BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS

PC-BSD 10.2

SYSTEM UPDATING & THE NEW PRIVACY/SECURITY OPTIONS

A SECURE WEBSERVER ON FREEBSD WITH HIAWATHA

PYTHON PASSIVE NETWORK MAPPING

THE LUMINA DESKTOP ENVIRONMENT

INTERVIEW WITH DRU LAVIGNE

VOL.9 NO.09
ISSUE 73
1898-9144

FREENAS MINI
STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

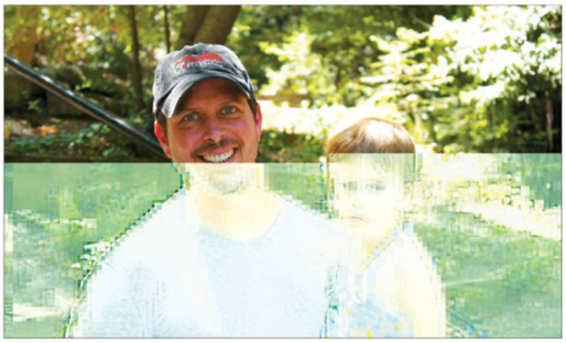
HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**



Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured

iXsystems™

http://www.iXsystems.com/mini

intel® inside™ ATOM™

FREENAS CERTIFIED STORAGE

With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

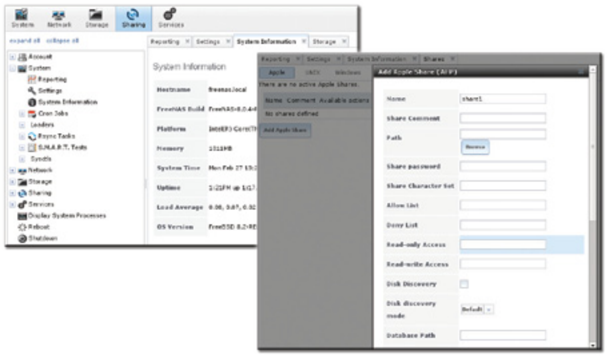
MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

» Custom built and optimized for your use case
» Installed, configured, tested, and guaranteed to work out of the box
» Supported by the Silicon Valley team that designed and built it
» Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

http://www.iXsystems.com/storage/freenas-certified-storage/

Dear Readers,

I think that we can say that we are happy to see all the changes and improvements in the latest version of FreeBSD. There are many new features that were added make our lives even easier every day. Let's see what the current version of FreeBSD and related products have to offer.

In this issue, we have gathered articles to give you deeper insight into the FreeBSD world. I recommend that you read the article written by Kris Moore about the New PC-BSD 10.2. Kris presents the newest PC-BSD features and fixes pushed into the latest version, 10.2. Some of these changes are large, while others occur behind the scenes to improve the PC-BSD experience. Among these changes are a brand new updating scheme, changes to disk partitioning with support for EFI, new privacy & security utilities and a new remote accessible AppCafe. So let's read this article to find out more about the new PC-BSD release.

Our regular writer David Carlier presents an article on creating a secure webserver on FreeBSD with Hiawatha. Hiawatha is a distinctive webserver application renowned for it's solid security and well audited code. Apart from having CGI/FastCGI support, SSL, Ipv6, Virtual hosts, it also protects against SQL injection and XSS, CSRF natively. David tells you more about it and presents how to deal with it in his article.

I would like to mention our article on the Lumina Desktop Environment by Ken Moore that is a new, BSD-licensed, graphical system environment, which is designed primarily for BSD and UNIX-based operating systems. In this article, you will learn a bit of the history behind the Lumina desktop, the foundations of it's design philosophy, the internal operating system support framework, the current status of the project, and the goals/ timeline for the project to achieve an official (non-beta) release.

Finally, please read the column by Rob Somerville. This time Rob shares more about his thoughts on emerging technology and his opinion on how it can be a force for both good and evil. To see the entire information about the BSD issue, especially the Interview with Dru Lavigne and the new article published in the Expert says section, I cordially invite you to check the next pages and review the Table of Content for this issue.

As always, I want to thank you all for your great work and help.

Enjoy reading!
Ewa & BSD Team

MAGAZINE BSD

Editor in Chief:
Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:
Michael Shirk, Andrey Vedikhin, Petr Topiarz, Solène Rapenne, Anton Borisov, Jeroen van Nieuwenhuizen, José B. Alós, Luke Marsden, Salih Khan, Arkadiusz Majewski, BEng, Toki Winter, Wesley Mouedine Assaby, Rob Somerville

Top Betatesters & Proofreaders:
Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman, Mark VonFange

Special Thanks:
Annie Zhang
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:
Paweł Marciniak
pawel@software.com.pl

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Publisher:
Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

ServerU

Rack-mount networking server

Designed for BSD and Linux Systems

DESIGNED FOR FreeBSD | DESIGNED FOR GNU / Linux | DESIGNED FOR PRO apps FreeBSD Enterprise Appliance | DESIGNED FOR pfSense

Designed. Certified. Supported

Up to **5.5Gbit/s** routing power!

KEY FEATURES

- 6 NICs w/ Intel igb(4) driver w/ bypass
- Hand-picked server chipsets
- Netmap Ready (FreeBSD & pfSense)
- Up to 14 Gigabit expansion ports
- Up to 4x10GbE SFP+ expansion

PERFECT FOR

- BGP & OSPF routing
- Firewall & UTM Security Appliances
- Intrusion Detection & WAF
- CDN & Web Cache / Proxy
- E-mail Server & SMTP Filtering

contactus@serveru.us | www.serveru.us
8001 NW 64th St. Miami, LF 33166 | +1 (305) 421-9956

Attend

InterDrone

The International Drone Conference and Exposition

InterDrone is Three Awesome Conferences:

Drone TECHCON
For Builders
More than 35 classes, tutorials and panels for hardware and embedded engineers, designers and software developers building commercial drones and the software that controls them.

Drone FLYER
For Flyers and Buyers
More than 35 tutorials and classes on drone operations, flying tips and tricks, range, navigation, payloads, stability, avoiding crashes, power, environmental considerations, which drone is for you, and more!

Drone BUSINESS
For Business Owners, Entrepreneurs & Dealers
Classes will focus on running a drone business, the latest FAA requirements and restrictions, supporting and educating drone buyers, marketing drone services, and where the next hot opportunities are likely to be!

The Largest Commercial Drone Show in North America

Meet with 80+ exhibitors!
Demos! Panels! Keynotes!
The Zipline!

September 9-10-11, 2015
Rio, Las Vegas
www.InterDrone.com

A BZ Media Event

ASUSTOR Released the AS1004T

Now on the market, the new 4-Bay NAS case from ASUSTOR, the AS1004T. It supports Windows XP, Vista, 7, 8, Server 2003, Server 2008, Server 2012, Mac OS X 10.6 or later, UNIX, LINUX and BSD operating systems. It was built with hardware encryption engine (256-bit AES military grade encryption for each folder), 2x USB 3.0 ports, 1x Gigabit Ethernet and a 120mm quiet cooling fan. The AS1004T was designed for offices and home users. AS10 series devices are equipped with dual-core processors, Gigabit Ethernet interfaces, and come with ASUSTOR's dedicated ADM operating system. They provide read speeds of over 110 MB/s and write speeds of over 96 MB/s, redefining the price to performance ratio of an entry level NAS.

About ASUSTOR. Founded in 2011, ASUSTOR Inc., a subsidiary of ASUSTeK Computer Inc., is a leading innovator and provider of private cloud storage (network attached storage) and video surveillance (network video recorder) solutions. ASUSTOR is devoted to providing the world with unparalleled user experiences and the most complete set of network storage solutions possible.

http://www.asustor.com/

iXsystems Announces Availability of FreeNAS 9.3 for General Consumption

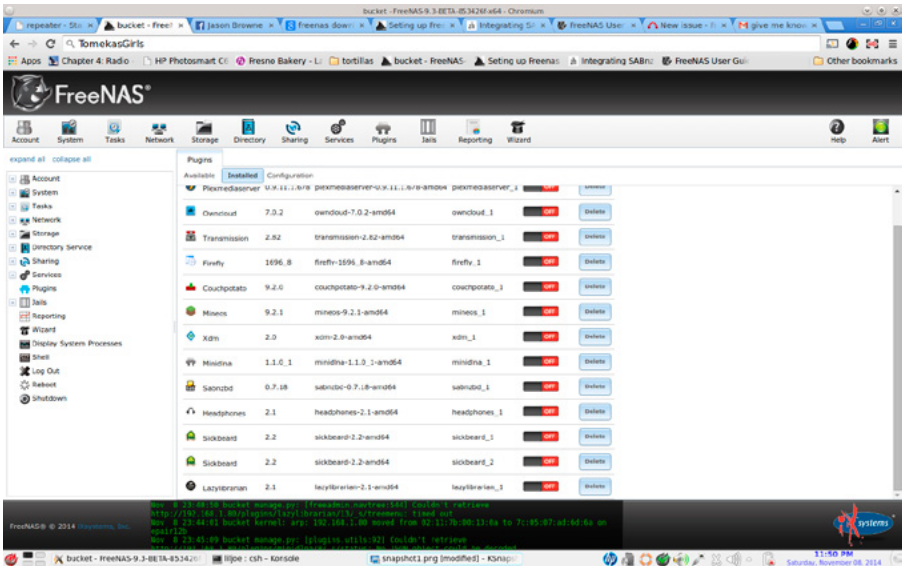
iXsystems announced the general availability of FreeNAS version 9.3. Highlights of the release include a simplified and revamped Web UI, automated updates, a new set-up wizard, and ZFS boot environments. FreeNAS 9.3 underwent extensive testing by thousands of BETA testers from the FreeNAS Community.

A change in the iSCSI target from userland to kernel space unlocked several of the newest block storage features and yielded far better performance than the previous user-mode iSCSI target. FreeNAS 9.3 also now supports coherent VMware snapshots, so ZFS snapshots and VMware snapshots are properly coordinated. Customers moving VMs will not have to move the snapshots as they are kept with the VM image on the FreeNAS server.

iXsystems worked with FreeBSD developers to add additional VMware VAAI primitives to FreeBSD in order to improve VMware support in FreeNAS. The VAAI block primitives, as well as the thin provisioning primitives, are now fully supported. This and other changes in FreeNAS 9.3 mean FreeNAS is significantly more viable as backing storage for VMware deployments.

Additionally, FreeNAS 9.3 adds support for Cluster Shared Volumes (CSV), Windows 2008 and 2012 R2 clustering, and Microsoft ODX Acceleration. FreeNAS users around the world have been using the in-kernel iSCSI (formerly a default option) as the block target in production for several months. Both external and internal testing show excellent stability as well as general performance improvements.

The FreeNAS boot device now uses ZFS, and boots with GRUB to allow administrators to boot from a clone of a previous boot environment and roll-back any undesired



Plexmediaserver	0.9.11.1.678	plexmediaserver-0.9.11.1.678-amd64	plexmediaserver_1	OFF	Delete
Owncloud	7.0.2	owncloud-7.0.2-amd64	owncloud_1	OFF	Delete
Transmission	2.82	transmission-2.82-amd64	transmission_1	OFF	Delete
Firefly	1696_8	firefly-1696_8-amd64	firefly_1	OFF	Delete
Couchpotato	9.2.0	couchpotato-9.2.0-amd64	couchpotato_1	OFF	Delete
Mineos	9.2.1	mineos-9.2.1-amd64	mineos_1	OFF	Delete
Xdm	2.0	xdm-2.0-amd64	xdm_1	OFF	Delete
Minidna	1.1.0_1	minidna-1.1.0_1-amd64	minidna_1	OFF	Delete
Sabnzbd	0.7.18	sabnzbd-0.7.18-amd64	sabnzbd_1	OFF	Delete
Headphones	2.1	headphones-2.1-amd64	headphones_1	OFF	Delete
Sickbeard	2.2	sickbeard-2.2-amd64	sickbeard_1	OFF	Delete
Sickbeard	2.2	sickbeard-2.2-amd64	sickbeard_2	OFF	Delete
Lazylibrarian	2.1	lazylibrarian-2.1-amd64	lazylibrarian_1	OFF	Delete

```
Nov  8 23:40:50 bucket manage.py: [freeadmin.navtree:544] Couldn't retrieve
http://192.168.1.80/plugins/lazylibrarian/13/_s/treemenu: timed out
Nov  8 23:44:01 bucket kernel: arp: 192.168.1.80 moved from 02:11:7b:00:13:0a to 7c:05:07:ad:6d:6a on
epair12b
Nov  8 23:45:09 bucket manage.py: [plugins.utils:92] Couldn't retrieve
```

changes. These technology improvements open up several new administrative options, such as maintaining copies of boot environments with previous firmware revisions and maintaining separate environments on a single appliance.

https://www.ixsystems.com/whats-new/ixsystems-announces-availability-of-freenas-9-3-for-general-consumption/

Mellanox OFED for FreeBSD

Clusters using commodity servers and storage systems are seeing widespread deployments in large and growing markets such as high performance computing, data warehousing, online transaction processing, financial services and large scale web 2.0 deployments. To enable distributed computing transparently with maximum efficiency, applications in these markets require the highest I/O bandwidth and lowest possible latency. These requirements are compounded with the need to support a large interoperable ecosystem of networking, storage, and other applications and interfaces. Mellanox offers a robust and full set of protocol software and driver for FreeBSD with the ConnectX®-3 / ConnectX®-3 Pro Host Adapters with Ethernet, InfiniBand and RoCE.

Main Features

- High performance single/Dual port support with full line rate, full duplex FDR InfiniBand and up to 40GbE
- RDMA and RoCE (RDMA Over Converged Ethernet) Transport offload with zero copy for low CPU utilization
- TCP/IP stateless offload, and Hardware Checksum Offload for Tx and Rx packets
- Net device statistics per port with both ports in active mode
- Extensive VLAN support with VLAN Tx/Rx acceleration (Hardware VLAN stripping/insertion), Hardware VLAN filtering and Hardware multicast filtering
- Adaptive Interrupt moderation on the receive path patterned according to either latency-bound traffic or bandwidth bound traffic
- Use fewer high-performance systems in a rack based system, reducing cabling, low power consumption
- Supports FreeBSD 10.0 and above

http://www.mellanox.com/page/products_dyn?product_family=193&

Mellanox® TECHNOLOGIES

Connect. Accelerate. Outperform.™

The Design and Implementation of the FreeBSD Operating System 2nd Edition

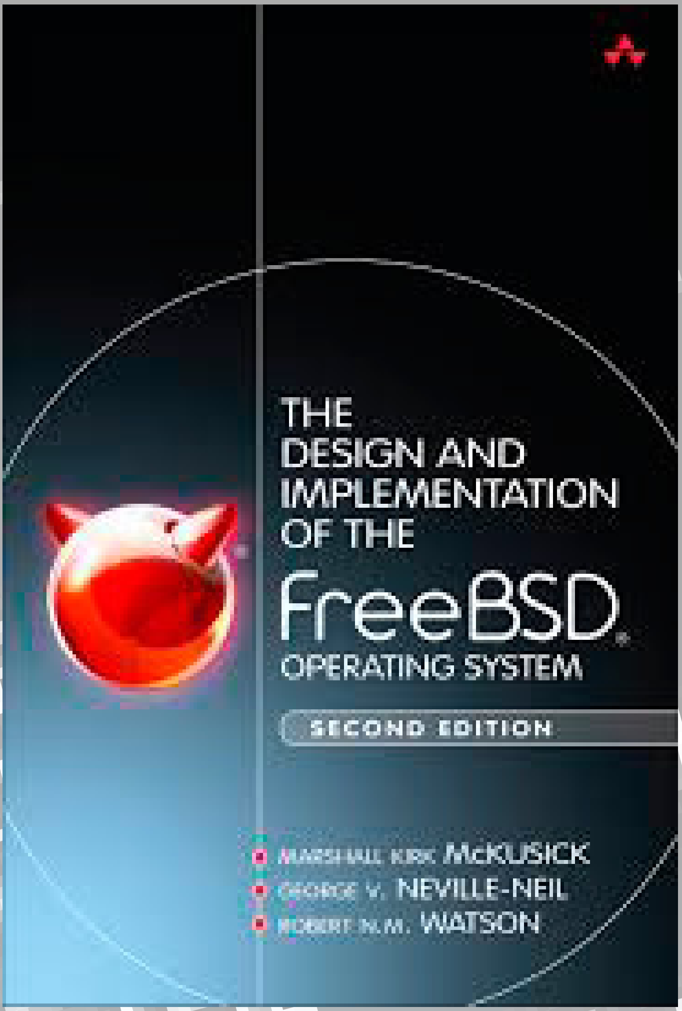
This book is the technical guide to the FreeBSD kernel's internal structure which has now been extensively updated to cover all major improvements between Versions 5 and 11. Approximately one-third of this edition's content is completely new, and another one-third has been extensively rewritten.

This Second Edition of the book:

- Explains highly scalable and lightweight virtualization using FreeBSD jails, and virtual-machine acceleration with Xen and Virtio device paravirtualization
- Describes new security features such as Capsicum sandboxing and GELI cryptographic disk protection
- Fully covers NFSv4 and Open Solaris ZFS support
- Introduces FreeBSD's enhanced volume management and new journaled soft updates
- Explains DTrace's fine-grained process debugging/profiling
- Reflects major improvements to networking, wireless, and USB support

This book can be used as both a working reference and an in-depth study of a leading contemporary, portable, open source operating system. Professionals will discover both FreeBSD's capabilities and its limitations. You will learn how to effectively and efficiently interface with it; how to maintain, tune, and configure it; and systems programmers will learn how to extend, enhance, and interface with it.

http://www.informit.com/store/design-and-implementation-of-the-freebsd-operating-9780321968975



by Marshall Kirk McKusick

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES
1.925.240.6652

$39.95
FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

$29.95
PC-BSD 9.1 DVD

Save a BUNDLE!

$49.95
The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

$99.95
The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

Stylish Dress Attire
Look Your Professional Best

Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD $39.95

CD Set Contains:

Disc 1	Installation Boot LiveCD (i386)
Disc 2	Essential Packages Xorg (i386)
Disc 3	Essential Packages, GNOME2 (i386)
Disc 4	Essential Packages (i386)

FreeBSD 9.0 CD $39.95
FreeBSD 9.0 DVD $39.95

FreeBSD Subscriptions

Save time and $$$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 .... $29.95
FreeBSD Subscription, start with DVD 9.1 .... $29.95
FreeBSD Subscription, start with CD 9.0 .... $29.95
FreeBSD Subscription, start with DVD 9.0 .... $29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD $29.95
PC-BSD Subscription $19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) $39.95
The FreeBSD Handbook, Volume 2 (Admin Guide) $39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) $59.95
The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 .... $79.95

PC-BSD 9.0 Users Handbook $24.95

BSD Magazine $11.99

The FreeBSD Toolkit DVD $39.95

FreeBSD Mousepad $10.00

FreeBSD & PCBSD Caps $20.00

BSD Daemon Horns $2.00

Bundle Specials!
Save $$$

Just Plain Fun
Mousepads & Novelty Horns

BSD Magazine
Available Monthly

FreeBSD Mall

For even **MORE** items visit our website today!
www.FreeBSDMall.com

Course Plan:

Pre-Course Material

- Why Do I Need a Pen Test Lab
- Definitions
- Creating Directory Structure For the Course
- Download Virtual Images
- Acquire Nessus Licenses

Module 1 The Build

- Definitions
- Some Basic Linux Commands You Need to Know

Software

- Installation of VMPlayer and Virtual Box. You Decide, We Will Cover Both.
- Setup of Our Penetration Testing System – Kali Linux Distribution
- Setup a Linux Client as a Virtual Machine
- Setup Our First Vulnerable Machine Metasploitable2
- Setup Our Second Vulnerable Machine Bee-box (BWAMP)

Exercises

- Overview of Virtual Machine Settings
- Run the Basic Linux commands
- Upgrade Kali Linux Distribution

Module 2 Port Scanning

- Nmap and Zenmap Installation
- Nmap Basic Scanning
- ZenMap Basic Scanning
- Metasploitable Dnmap Scanning

Exercises

- Run Nmap Scans against Ubuntu
- Run Zenmap Scans Against Metasploitable2
- Run Dnmap Scans Against Host

Module 3 Vulnerability Scans

- Installation and Licensing of Nessus Vulnerability Scanner
- Installation of Netsparker Web Vulnerability Scanner
- Basic Nessus Scanning
- Basic Netsparker Scanning
- Intermediate Nmap Scans

Exercises

- Run a Nessus Scan Against Metasploitable2
- Run a Netsparker Scans Against Bee-Box (BWAMP)
- Run a Nessus Scan Against Ubuntu

Module 4 Advanced Scanning and Reporting

- Nessus Advanced Scans
- Netsparker Advanced Scans
- Nmap Advanced Scans
- Metasploit Reporting
- Review Other Resources Available to You…
- Where Do I Get Virtual Machines

Exercises

- Create a Metasploit Report Combining Nessus and Dnmap Scans
- Run an Advanced Nessus Scan Against Metasploitable 2
- Run an Advanced Netsparker Scan Against Bee-Box (BWAMP)

A Look at the New PC-BSD 10.2

BY KRIS MOORE

Since PC-BSD 10.1 was released in November 2014, there have been many exciting new features and fixes pushed into the latest version, 10.2. Some of these changes are large, while others occur behind the scenes to improve the PC-BSD experience. Among these changes are a brand new updating scheme, changes to disk partitioning with support for EFI, new privacy & security utilities and a new remote accessible AppCafe. In this article, we will take a closer look at two of these, system updating and the new privacy / security options in the form of PersonaCrypt.

What you will learn…

- The features of the new FreeBSD 10.2 release.
- How the system update and the new privacy / security options in the form of PersonaCrypt work.

What you should know…

- The previous releases of the FreeBSD OS.
- Some familiarity with various FreeBSD topics.

The first of the new features was a large update to the entire system updating process used by PC-BSD. Historically, PC-BSD has used the standard updating mechanisms in FreeBSD, such as ‚freebsd-update' and ‚pkg upgrade' in an automated fashion. One of the biggest pain-points, however, was the sheer complexity in dealing with packages and their related dependencies. While the recent updates to PKGNG have helped with the complexity, there were still too many failed upgrades for comfort, often preventing the PC-BSD tools from running in an automated fashion. To assist with this problem, PC-BSD had Boot-Environments functionality that allowed rolling back if something went wrong, but we still wanted to fix the underlying problems. To do so, we came up with a new mechanism for doing system updates that relies upon the power of ZFS/Boot-Environments and the existing FreeBSD tools.

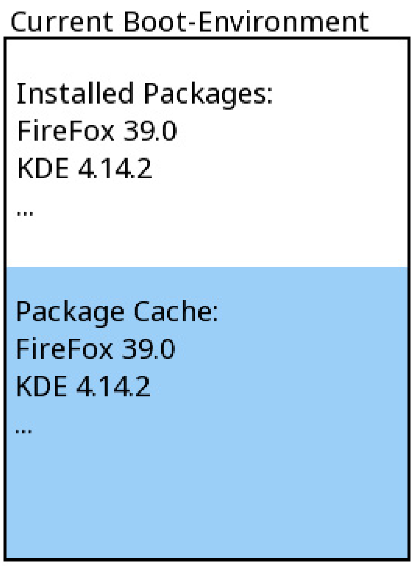


Figure 1. *System state before upgrade*

The new "update" process on PC-BSD works in the following manner. When performing an update of any packages, it first queries the local package database to determine the "top-level" packages, those installed by the user which nothing directly depends on. This list is saved as the master package list of the system and used immediately to begin the process of downloading the latest versions from the mirrors. These files are stored in a local cache on disk, which is cleaned of obsolete packages before any upgrade. Once the top-level packages and the related dependencies are downloaded, the update moves into its next phase. A new ZFS Boot-Environment is created and then prepared, first by cleaning out the old packages and then re-installing the top-level packages only. This helps bypass the issues which occur while trying to solve the conflicts between old and new dependencies of the top-level packages. Once the update is finished, the new BE is marked as "active" and the user is notified that they can reboot when they are ready to load into the updated system.

Using this method has proved advantageous in a number of ways. First, the currently running boot-environment is never touched. This means you can keep working and using a system without interruption and reboot only at your convenience, such as at the end of the day. Secondly, while failures are much less likely, they are also no longer going to interrupt the system. Updates can be re-run as many times as desired, with the new BE created and flagged as active at the finish. Lastly, package updates become much more reliable, since we are duplicating the process of installing packages fresh, where there are no packages on the box to cause conflicts in the first place. This has already proven to be a much more reliable upgrade process, even allowing us to do daily or weekly updates with ease.

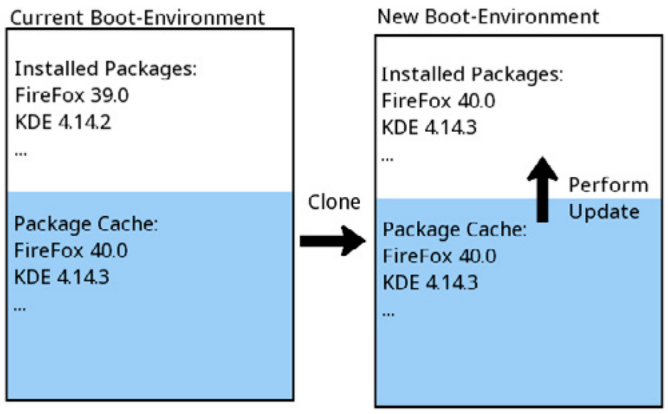


Figure 2. *Current BE is cloned and new packages installed*

In addition to the new updating methods offered in PC-BSD 10.2, there have also been major changes to improve both security and privacy. For the past several months, the entire package repository for PC-BSD has been switched to build against LibreSSL, instead of the default OpenSSL used by FreeBSD upstream. This change is almost entirely transparent to end-users, but offers a much smaller vulnerability target and often less critical security updates being dropped on us from upstream. 10.2 also brings with it the new utility "PersonaCrypt" and graphical options for system-level Tor proxy.

PersonaCrypt provides some unique methods to improve user privacy. It is integrated with the PC-BSD graphical login manager (PCDM) to do the following:

- Allow the users $HOME directory to be stored on an encrypted external media using GELI/ZFS
- Perform "Stealth" mode logins, where the $HOME directory is created on a GELI-backed one-time encrypted image with no personal information contained within

The first of these options can be used in a couple of interesting ways. It can be used either with a secondary internal disk to provide a separate encrypted $HOME directory for a single system, or it can be used with external media (such as a USB stick) to provide a portable $HOME directory. The key is automatically split into two parts, so that the first resides on the local system, while the second part is provided as a password at the login screen. PersonaCrypt allows the system key file to be exported from the initial system, and imported or "paired" with another system, such as in the case of a desktop and laptop. This means, should the external media be lost or stolen, it cannot be accessed without both the system it is paired to, and the user-password. It also can be used as a convenience option, where moving your entire $HOME directory between two desktops (such as home and office) can now be done in an offline and secure manner. With media using ZFS, it also means that snapshots and replication options can be used for backups either manually, or via PC-BSD's Life-Preserver utility.

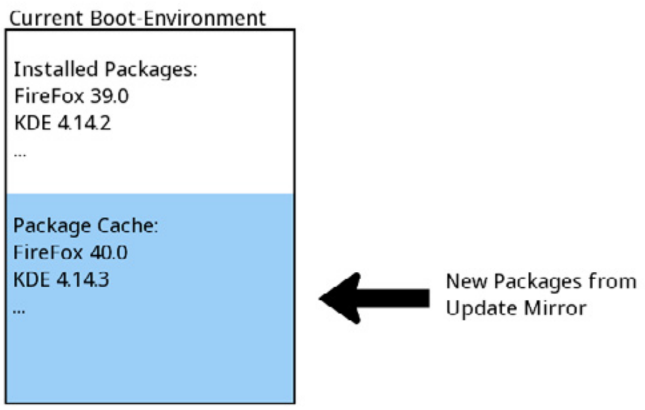


Figure 3. *Package updates being downloaded to local cache*

The "Stealth" mode option also uses GELI in a unique way to provide an "anonymous" desktop session. At the graphical login manager, the user can simply check the "Stealth" mode option, which takes the following steps:Creates a temporary ZFS ZVOL

1. Sets up a one-time GELI key on the ZVOL
2. Formats the GELI/ZVOL with UFS
3. Mounts the device on top of the user's $HOME directory
4. Copies the /usr/share/skel data to the new $HOME
5. Performs the usual login

This method ensures that no identifying data on the user is available on $HOME and as such provides a "blank-slate" for any apps which are run. When the user session is finished, the new ZVOL is destroyed and the old $HOME directory is visible again. Because we are using a GELI one-time key, if the system is rebooted or crashes without doing the cleanup, the contents on disk are still rendered useless. This mode can be particularly useful when run in conjunction with Tor mode on the system. By enabling Tor via the System Tray GUI, the system's firewall will be reset, and all DNS/TCP traffic is automatically re-directed through a Tor transparent proxy, even if the local applications do not directly support Tor / socks proxy mode.

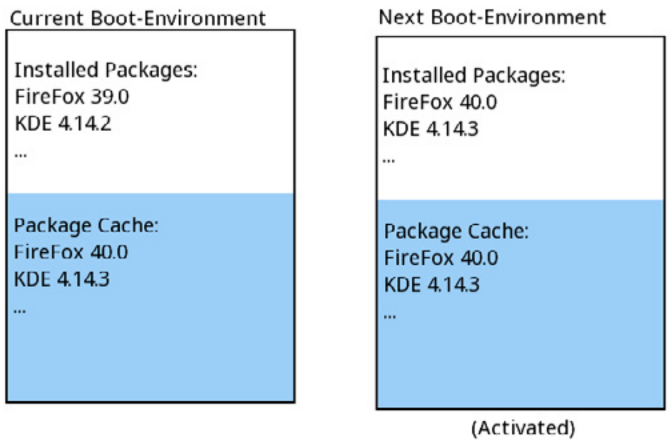


Figure 4. *New BE is marked as default / active at next reboot*

These are two components of PC-BSD that have changed greatly from version 10.1 to 10.2, and provide fairly unique functionality to end-users. There are also a slew of other features and fixes which have gone into 10.2, easily making it the best PC-BSD release yet. Users are more than welcome to join the conversation on the official PC-BSD forums, mailing lists, and IRC channel (#pcbsd on FreeNode).

ABOUT THE AUTHOR

Kris Moore is the founder and lead developer of PC-BSD. He lives with his wife and four children in East Tennessee (USA), and enjoys building custom PCs and gaming in his (limited) spare time. To contact the author, email him at kris@pcbsd.org

Among clouds Performance and Reliability is critical

Download syslog-ng Premium Edition product evaluation here

Attend to a free logging tech webinar here

BalaBit
IT Security
www.balabit.com

syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage

Basis Of The Lumina Desktop Environment

KEN MOORE

The Lumina Desktop Environment is a new, BSD-licensed, graphical system environment which is designed primarily for BSD and UNIX-based operating systems. This focus on BSD systems results in a number of distinct differences in from the current collection of Linux-focused desktop environments, only one of which is independence from all the Linux-based system management frameworks.

This article will explain a bit of the history behind the Lumina desktop, the foundations of its design philosophy, the internal operating system support framework, the current status of the project, and the goals/timeline for the project to achieve an official (non-beta) release.

Introduction/History

The Lumina Desktop Environment ("Lumina" for short), was initially started in late 2012 as a small hobby project to just add some functionality/utilities to the Fluxbox window manager. I had grown a bit tired of all the constant issues with running other desktop environments on FreeBSD. I was using Fluxbox quite a bit at that time because I liked the responsiveness of the interface – although I really missed things like automatically generated application lists/launchers and the ability to have the system remember/detect which application to use to open a file. After a little over a year of intermittent work on nights/weekends, I had gradually turned Lumina into a very basic Qt4-based graphical overlay for Fluxbox, with a small background utility for launching applications and opening files. Around this time (end of 2013, beginning of 2014) it was still considered barely functional (pre-0.1), but Kris Moore of the PC-BSD project offered to host it on the PC-BSD source repository, tying it into the automated build and distribution systems for users of the PC-BSD operating system to play with. Since we were in the middle of phasing out some old technologies that I maintained as part of my PC-BSD development job, I took him up on the offer and we moved the source tree for Lumina over to the PC-BSD SVN repository so I could work on it a bit more regularly as time availed itself. Over the next couple months, I was able to continue to clean things up and get it to a more usable state (although still very primitive), all while still maintaining a fairly tight level of secrecy on the project (as secret as an open source project on a public source repository can be, at least).

In April of 2014 that all changed: a status update about Lumina was accidentally featured in one of the PC-BSD weekly blog updates after we finally got a port and pre-built packages for it quietly added to the FreeBSD ports tree. This caused PC-BSD users to get all excited, some news sites and slashdot picked up on the story, and then the secret was officially out – the BSD's were getting their own desktop environment. Since then, a lot has changed in the project: the sources were moved from SVN to the new PC-BSD repository on GitHub in April 2014, then in September 2014, it was moved again to its own sub-repository on GitHub (under the PC-BSD umbrella) and also obtained an independent release engineering schedule. In December of 2014, the entire project was converted to Qt5 and parts of it also started getting converted from the Xlib to XCB libraries (since Qt5 used XCB, whereas Qt4 used Xlib). All the while, more and more of Fluxbox was getting disabled/replaced/bypassed all the time, so that in May of 2015, we announced that we were starting work on a new Qt-based window manager specifically to integrate with the Lumina desktop on a tighter level and replace Fluxbox. This leads us up to the present time: Lumina is still technically a beta release (since the new window manager has not been finished/implemented yet), is available on many operating systems (BSD and Linux-based), and is continuing to become more popular with every new point release. Why is this so? Perhaps it has to do with the philosophy behind how the desktop is designed and implemented...

Design Philosophy

The Lumina desktop is designed around three primary goals: minimal system overhead, a modular interface, and a complementary relationship with the underlying operating system. Minimal system overhead is an obvious goal – people use computers to run apps to get things done, not look at a pretty interface. Any system which uses most of its processing power and/or memory just to run the interface is a system which has stopped being a tool and just become another burden. In my experience, the main culprits for this type of system drain are automatically started background daemons that are either rarely accessed or are used to hide the actual impact/requirements of some client application. Lumina takes a very clear-cut approach with regard to system overhead: keep it simple. The entire Lumina desktop can be boiled down to a single library, one binary for the entire desktop session, and one utility for opening files and/or launching applications. The other various applications which Lumina provides (such as the file manager) are simply treated as optional convenience utilities to be used/ignored/removed as desired. Because of this simplicity, there is no need for any communications daemons (such as DBus) or other desktop-monitoring daemons just to keep the desktop working in a semi-unified fashion, allowing Lumina to run incredibly fast and with a tiny amount of system overhead (typically less than 150MB of memory).

The principle of a modular application is something that has been around for quite a while with varying degrees of success. The main way that this has been performed is through detection/use of optional "plugin" libraries (or other similar systems), and while this works well in theory, it also has the effect of also making your application highly sensitive to "bad" plugins (whether malicious or just incompatible). Since the system interface is one thing which should be rock solid and reliable, Lumina is designed around a system of built-in plugins – functionality that is always in sync with the current version of the desktop because it is compiled and included within the desktop binary itself (preventing both version mismatch issues and externally-loadable 3rd-party malware).

Now how does a desktop environment have anything but a complementary relationship with the underlying OS? Very simple. If you look at the Linux development space, you can easily see that there is no such thing as a "Linux" operating system. Instead, what you have is a Linux kernel combined with some userland to create a "distribution" of Linux, such as a GNU/Linux distribution. The Linux-based desktops which are developed in this ecosystem have gradually evolved into providing much of that userland "glue" to assemble all these various operating systems – effectively becoming an embedded part of the OS and providing many of the basic OS-level standards for application compatibility between different Linux distributions. While this seems to work fine in Linux-space (most of the time), it causes all sorts of issues when you try to run that desktop on a non-Linux system because the BSDs already have a completely functional operating system independent of any desktop environment and its underlying frameworks.

For a quick example, the background system frameworks which the linux-based desktops use many times will not run (or will not run well) on BSD-based systems because they are either trying to duplicate functionality which is already implemented in a different way on a BSD operating system (such as the *hald* vs *devd* issues) or they rely on functionality which is not available on a BSD system (at least not in the same way). By writing a desktop to rely on these frameworks then is to effectively limit the portability/functionality of the desktop across various OSs (and don't forget the extra overhead required for all the intermediate interface layers).

Operating System Integration

The Lumina desktop handles OS-integration in a very efficient, but compartmentalized manner. First, Lumina restricts itself to being a system interface only and does not try to provide utilities for configuring every aspect of the system. The reason for this is quite simple – not all systems are used in the same situations or for the same purpose. For example: a stand-alone service kiosk does not need to have access to all the same configuration utilities that a system developer might need on his workstation. Similarly, an embedded system with a tiny screen (such as a cell phone), will need utilities tailored for touch-screen use while a console television system will needs apps tailored to function with a simple controller. By keeping Lumina as a customizable interface only, it allows the distributor of the system to decide what types of apps need to be installed based on their target audience and system specifications. For an example of how this works, look at the relationship between Lumina and the PC-BSD project. PC-BSD provides a FreeBSD operating system tailored for desktop/laptop users and also provides all the tools and utilities necessary for configuring that system. Lumina just provides an OS-agnostic system interface with the ability to embed support for all those various OS utilities (control panel, package manager, etc.) into the interface so that they are readily accessible by the user.

Second, for controls which an interface needs to have access to but the OS manages, Lumina has a single *LuminaOS* class contained in its library where all those interactions may be set to directly interact with the native system/libraries. A couple good examples of this would be controlling screen brightness or audio volume. The OS and/or distributor has usually already decided which audio subsystem will be used (*ALSA*, *OSS*, *PulseAudio*, etc) – Lumina just needs to know how to have basic interactions with it. By the same token, though, all those OS-interactions are also completely optional. In the class, it is very simple to specify when something is not available for an OS, and in that case the interface responds by hiding all references to that functionality – preventing the user from having access to some of the niceties that Lumina provides, but still allowing the user to have a fully-functional interface.

Distribution Framework

Since Lumina is designed around a distribution framework which utilizes a distributor before it reaches the consumer, this opens up all sorts of configuration/implementation possibilities that Lumina is only just starting to expose. First, this opens the door for Lumina to be used on almost any type of device: such as a smartphone, tablet, or console tv system, in additional to the traditional “desktop” market. This is due in part to the compartmentalization of all the interface elements into distinct “plugins” – since this allows the distributor to pre-configure the desktop with an interface suited for their target device (just as PC-BSD is doing for desktop systems right now). Second, this allows the user the ability to suit their workflow and maximize productivity (such as having the distributor set a default interface that caters to the target user). Third, this allows the desktop to be as heavy/light on resources as desired (either for system performance or user preference), depending on the choice of interface elements and desktop functionality.

Customization options like this always come with a price: complexity and/or confusion. Lumina attempts to alleviate this in a couple different ways. By providing a single settings file on the system which arranges the default user interface and other user-experience options, Lumina makes it extremely simple for distributors and system administrators to customize the out-of-box experience, as well as by providing a convenience utility called *lumina-config* which gives the end-user an easy way to configure every aspect of the interface as desired (all without touching a terminal or text editor).

Future Goals

Right now, the primary development focus for Lumina is to achieve a comparable level of functionality with other common desktop environments (particularly with regards to traditional desktop systems – mobile/console systems are not a huge focus right now), and finally reach the goal of the first non-development release (1.0). The means that the new Qt-based window manager needs to get finished up (with a current target of December 2015) as well as the continued addition of new interface plugins and features. This new window manager is actually three-fold, it will not only replace Fluxbox for managing windows while the session is active, but it will also provide a replacement for xscreensaver while the session is inactive and also incorporate the session locking/suspend functionality for when the user needs to step away from the system. This utility, when combined with the current desktop interface, will effectively make Lumina a stand-alone system interface with an incredibly small list of system dependencies – opening up all sorts of possibilities for system types and without limiting the user by requiring long lists of package/library dependencies which may conflict with what the user actually needs the system to do. The current goal for reaching version 1.0 of the Lumina desktop is July of 2016 (prior to FreeBSD 11.0-Release), and so far we seem to be on track to meet this goal. In the meantime, the Lumina desktop is a highly fast/stable alternative to other common desktop environments, particularly for BSD-based systems. The only time we expect that a user’s current settings may get changed on an upgrade in the near future is when the new window manager gets activated by default (simply due to the number of configurations/files that this change touches). We are already working on ways to minimize the effect of these changes as much as possible,however, so that Lumina may continue to provide a stable, consistent user experience even across major updates.

ABOUT THE AUTHOR

Ken Moore is the founder and lead developer of the Lumina desktop environment as well as a developer with the PC-BSD project. He is an avid proponent of keeping it simple, and always focuses on clean, readable code quality as well as interface designs that anyone can understand and use. On the weekends, he likes to spend time with his wife and two kids, playing games, building contraptions, or just having general family time.

Learn How To Master Big Data

Big Data TECHCON

November 2-4, 2015

CHICAGO

Holiday Inn Chicago Mart Plaza River North

Choose from 55+ classes and tutorials!

Attend Big Data TechCon to get practical training on Hadoop, Spark, YARN, R, HBase, Hive, Predictive Analytics, and much more!

Take a Big Data analytics tutorial, dive deep into machine learning and NoSQL, learn how to master MongoDB and Cassandra, discover best practices for using graph databases such as Neo4j and more. You'll get the best Big Data training at Big Data TechCon!

www.BigDataTechCon.com

People are talking about BigData TechCon!

Great for quickly coming up to speed in the big data landscape.
—Ben Pollitt, Database Enginee, General Electric

There was a large quantity and variety of educational talks with very few sales lectures. It was just informative and inspiring. This was the best conference ever! Get a ticket for 2015!
—Byron Dover, Big Data Engineer, Rubicon Project

A BZ Media Event

A Secure Webserver on FreeBSD with Hiawatha

DAVID CARLIER

In most cases, when it comes to choosing a web server, Nginx quickly comes to mind (I personally appreciate this one a lot, no doubt about this). However, there exists an interesting alternative, which embeds some nice features, called Hiawatha.

What you will learn…

- The features of Hiawatha.
- How to configure the secure webserver.

What you should know…

- The basics of FreeBSD OS.
- Some familiarity with various security topics.

What makes Hiawatha special? First, its code is well audited and famous for its solidness in terms of security. Apart from having CGI/Fast-CGI support (hence possibly making dynamic websites with PHP-fpm), SSL, Ipv6, virtual hosts. It also provides protection against SQL injection and XSS, CSRF natively. It might lack support of third party modules, as the architecture does not allow it, but Hiawatha has reverse proxy support!

Luckily, FreeBSD already has a package/port, so once installed...

Configuration

Let's configure it. Here is a sample configuration, a FastCGI one. You can see that it appears very readable:

```
ErrorHandler = 404:/404.html

Binding {
        Port = 80
        Interface = ::1
        MaxKeepAlive = 30
    # Two first secure measures, limiting the client
   request size and the max time for
     # a client's connection to be kept open        MaxRe-
   questSize = 512
        TimeForRequest = 3,20
}
...
Binding {
        Port = 443
        Interface = ::1
        ...
        SSLcertfile = hiawata.pem
        RequireTLS = yes
        # Add X-Random header with a length of 256
        RandomHeader = 256
}
...
# A feature to make decisions based on url regexes
UrlToolkit {
        ToolkitID = phprewrite
         Method GET
         Match /private DenyAccess
         Match ^/page/(.*) Rewrite /index.php?page=$1
}
```

```
FastCGIServer {
        FastCGIid = PHP5
        UseToolkit = phprewrite
        # Can be the path to the unix socket too
        ConnectTo = 127.0.0.1:2005
        Extension = php
}

VirtualHost {
   Hostname = www.example.com
           WebsiteRoot = /usr/local/www/hiawatha
           AccessLogfile = /var/log/hiowatha/example-
   access.log
           ErrorLogfile = /var/log/hiowatha/example-error.
   log
           # Althought those directives can provide pro-
   tection, they are of course
     # not 100% reliable
           PreventCSRF = yes
           PreventSQLI = yes
           PreventXSS = yes
           # A body which matches this regex is forbidden
           DenyBody = ^.*%3Cscript.*%3C%2Fscript%3E.*$
           UseFastCGI = PHP5
           TimeForCGI = 5
           # By default it is index.html
           StartFile = index.php
}
```

So, for the moment, nothing special that hasn't been seen before. Let's dig in deeper. With Hiawatha we can set some banning policies like below:

```
# If in our VirtualHost above a user had
bad body content
BanOnDeniedBody = 300
  # If too many malformed HTTP requests
    are made by a client, it's banned for
      1 minute
        BanOnGarbage = 60
          # Banned if the request size
          exceeds this size
           BanOnMaxReqSize = 512
            # Anti flood measure; here, if
             the client does more than 20
             requests per second, it's
             # banned for 1 minute
             BadOnFlooding = 20/1:60
            BanOnSQLI = 60
           KickonBan = yes
          # Reset the ban times if the
   client attempts to connect when it is already banned
RebanDuringBan = yes
GarbageLogFile = /var/log/hiawatha/garbage.log
```

Also, Hiawatha provides a separate feature called Monitor; a feature a bit "a la" Munin but well adapted to this server.

```
# This IP address will be allowed to download the event
   log files for its analytics
MonitorServer = 192.168.1.2
```

As mentioned before, Hiawatha does not support third party modules, so if you wish for your service making decision to be based on country per IP with GeoIP for example, you can use Hiawatha as a Reverse Proxy in front of Nginx. Simply enter as shown below inside your VirtualHost setting:

```
VirtualHost {
      Hostname = www.myreverseproxy.com
      PreventCSRF = yes
      PreventSQLI = yes
      PreventXSS = yes
      ReverseProxy .* http://192.168.1.2:8111/geoip
}

# Cache internally those contents from reverse proxy
   requests per extension
CacheRProxyExtensions = gif,png,jpeg,css
CacheSize = 256
...
```

Conclusion

Hopefully, this article will give you the curiosity to consider this approach. For what it provides it is impressive, considering the fact that it is a one project maintained since 2002. (by Hugo Leisink)

ABOUT THE AUTHOR

David Carlier has been working as a software developer since 2001. He used FreeBSD for more than 10 years and starting this year, he became involved with the HardenedBSD project and performed serious developments on FreeBSD. He worked for a mobile product company that provides C++ APIs for two years in Ireland. From this, he became completely inspired to develop on FreeBSD.

Defeating CryptoLocker Attacks with ZFS

MICHAEL DEXTER

Plextec is a Canadian managed services provider that uses FreeNAS exclusively to provide Windows and GNU/Linux virtual servers to over 200 companies using XenServer. I spoke with Plextec CTO Todd Ladouceur about how Plextec routinely defeats CryptoLocker ransomware attacks with ZFS and FreeNAS.

Michael: Todd, what are CryptoLocker attacks?

Todd: CryptoLocker attacks are a category of clever yet nefarious personal computer malware that infects a PC via a tantalizing email message or link and silently encrypts your local disks and any network shares you are connected to. When finished encrypting, the malware holds your data for ransom, giving you on average three days to make a decision between paying the ransom or having your data destroyed forever. Organizations of all sizes have been hit by these "ransomware" attacks including police departments and hospitals and an early estimate put the damages at $30 million. The worst situation we have seen was when a user got infected on a Friday afternoon while catching up on email and dreaming about the weekend. The CryptoLocker malware took hold and had all weekend to encrypt every network share their system was connected to plus their local drive, wreaking havoc across the organization.

iXsystems™

Michael: Can you stop CryptoLocker attacks with antivirus software?

Just about every antivirus vendor has a fix for the various CryptoLocker attacks but they simply can't keep up with how quickly CryptoLocker attacks evolve. The organizations behind them are obviously well-funded and because the malware uses encryption, removing it does nothing to restore your data. In fact, to remove the malware could result in the instant loss of all your data because it is the one known tool that can decrypt it. Unfortunately, many CryptoLocker attacks attempt to destroy your backups on services like DropBox or in Windows Shadow Copies.

Michael: What role does ZFS play in combating CryptoLocker attacks?

Todd: We share FreeNAS-backed virtual machine images to our XenServer hosts over NFS and snapshot each VM's dataset on a 30 minute and hourly basis with a retention of one week for the 30 minute ones and one month for the hourly ones. We then replicate these snapshots to one or two additional FreeNAS servers. When a virtual machine is hit with CryptoLocker, we step through the snapshots on one of the replica systems until we find a point in time just before the attack. We clone the known-good snapshot and share it back to XenServer. We make sure the VM passes all of our quality checks and performs as expected, and then copy it back to the primary server through the XenCenter. We could just roll back the primary system but this strategy allows us to preserve the compromised VM for a few days for forensic purposes.

Michael: How long does the restoration process take?

Todd: On average we can get a Windows server back in production with full validation in under two hours. In a pinch we could simply roll back the primary server but we prefer maintain that extra layer of accountability. With ZFS we know our replicas are bit-for-bit identical to the originals so we do not hesitate in relying on them. A recovery from tape or an online provider would cost a fortune in time, money or both, and would not provide the assurances that ZFS gives us.

Michael: Are CryptoLocker attacks common?

Todd: They are way too common. We have had over ten clients hit with CryptoLocker malware and some of them multiple times. Some would easily be out of business because of it and I hate to think what would happen to us as their IT provider. The threat is real and ever evolving. We constantly revise how we can recover from CryptoLocker attacks more quickly and also educate our clients about how to protect themselves from these and other attacks. We have read about blocking CryptoLocker attacks with group policies and administrative controls but there is no way these steps can keep up with the ever-evolving threat.

Michael: Do you think FreeNAS and TrueNAS are safe from CryptoLocker attacks?

Todd: Absolutely. CryptoLocker attacks work on the file level rather than the block level, keeping our virtual machine images immune as long as you snapshot them regularly and retain enough snapshots to return to a point in time before the attack. To be vulnerable you would have to share your whole VM store over NFS to a compromised Windows client but even then the snapshots would still bring you back to safety because they are at the block level.

Basically, CryptoLocker is a joke with ZFS.

Todd Ladouceur
CTO, Plextec

For more information on FreeNAS Certified and TrueNAS storage systems, visit www.ixsystems.com or call 1-855-GREP-4-IX.

Emerging Technology Has Increasingly Been a Force for Both Good and Evil

ROB SOMERVILLE

Since the Second World War, emerging technology has increasingly been a force for both good and evil. The race for technological advance since then – while creating a level playing field for many – has still to address more fundamental inequalities and lead us into the Utopian dream. With transformational advances now being made so frequently, are we about to cross the Rubicon?

I am not long back from my annual vacation, and one of my everlasting memories of this year's break will be the encounter I had with a large black inanimate object. In hindsight, it would probably be more accurate to describe my encounter as being with a very zoetic object. The technology in question? The stunning and revolutionary Tesla Model S electric car. And I use the term "car" very loosely indeed. Most cars are utilitarian, but the Tesla is like a Breitling watch, or a Waterford crystal vase. Quality, beauty, functionality, impeccable design, and manners, all rolled into one package. What makes the Tesla unique though is not just the advanced technology that is the underpinnings of the the vehicle, but the synergy between vision and ethics. While there are arguments about the rare metals used in a lot of technology and the depletion of the Earth's resources, it would be harsh to condemn the designers for at least taking a step in the right direction: away from the internal combustion engine. Rome was not built in a day, and based on the innovation and ingenuity Tesla have shown so far, I can see no reason why they would not aggressively adopt different energy models and when they become feasible and available. But this is a distraction from the main event, the Model S, while occupying the moral high ground, is enormous fun. With a zero-to-sixty mph acceleration of under three seconds, any journey is concluded with the driver and passengers climbing out of the luxurious interior with a grin stretching from ear to ear. Some would class the Tesla as a rich man's plaything, but with the widespread adoption of hybrid solutions the direction of the automotive industry is clear.

I have been thinking about this encounter a lot. I'll be the first to admit, I am somewhat smitten, but my love for quality engineering has been renewed once again. While I have a few pipe dreams as to what I would do should I win the lottery, very few of these involved purchasing a luxury car. The Tesla has changed all that. Like a cute kitten or adorable puppy it has wormed its way into my heart in a way few other cars have – and that includes a smattering of Lamborghinis, Jaguars, Porches, TVR's and BMW's I have encountered – apart from my loyal metallic grey Saab 900 that I had to let go many years ago due to old age and overwork. A good friend, come rain shine or London traffic, Saab served me faithfully through the years. As the old adage goes, they don't make, them like they used to, although I have high hopes that Tesla will occupy the space vacated by the innovative Swedish manufacturer when they ceased trading a few years ago.

So from a purely personal standpoint, the only real moral danger I can see from the Tesla is a large number of flattened pedestrians as the car is totally silent. This is hardly even any tyre noise on the road when accelerating, a remarkable feat. The only reason that Tesla has not adopted some sort of external engine noise is that there is no official standard yet in Europe, but no doubt some committee somewhere will come up with some noisy discordant compromise that will detract from the overall feel.

On the other hand, there is considerable moral danger in other technology spheres. The ethics of self piloting cars aside, the futurists Vernor Vinge, Ray Kurzweil and others have warned for many years, that we are approaching a singularity where technology will surpass human capabilities. The irony here is that exponential increase in capacity does not rest at the hardware layer as encompassed by the well known Moore's law, but rather in the efficiency of the algorithms used. Professor Martin Grötschel estimates that this increase in efficiency is forty-three times that of the hardware, according to Kurtzell. That in itself should not be of particular concern, other than we need to keep a watchful eye and remember that technology is our slave and not our master. I firmly believe that while there is a natural firewall between mankind and technology, whether that be via the "Off switch" or indeed just simple choice, we are relatively safe. The danger becomes critical where systemically man cannot detach himself from the machine. From a cognitive perspective, the heady mix of the Internet, World Wide Web and smartphones is already blurring the boundaries. As an adult, I can look at this rationally and detach myself without too much pain from my Android device. For the younger (and some of the older) generation, this is a major problem. So personally I would be very wary indeed of where we are going in the area of transhumanism, especially when some of the potential visions for the future include nano-robots are crawling around inside our brains. To quote Francis Fukuyama, the greatest threat to humanity are those whose vision is to "liberate the human race from its biological constraints".

A colleague of mine jokingly commented how I always manage to slip politics into my writing, and the edition of this column is no exception. In 2016, Zoltan Istvan, of the Transhumanism party, will be running for US president. While he openly admits, there is very little chance of winning the popular consensus, the transhumanists hope to become more prominent over the next election cycles. With extensive media coverage, we are already heading towards a clash of cultures where the result of ethical and moral argument will be critical. Herbert Hoover in 1928 offered Americans a "Chicken in every pot". In 2016, Zoltan Istvan will offer not just chickens, but human immortality. Indeed, many scientists suggest this could be accomplished in the next 20-30 years, possibly sooner if there is enough political and financial will.

The real world consequences of transhumanism are probably greater than the discoveries surrounding the splitting of the atom, quantum physics, penicillin, birth control, the printing press and the Internet combined. Zoltan Istvan is to be applauded for having the courage of his convictions bringing debate about this imminent quantum leap to the fore and into the broader sphere of national consciousness. He makes a lot of very valid points. This, however, leaves us technologists with a big problem. We can no longer hide behind our monitors or compilers. We are now part of a much larger landscape, the political, moral, ethical and spiritual agenda. Questions will be asked of us, and we will need to search our hearts very deeply indeed if we are to give an honest answer.

ABOUT THE AUTHOR

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Interview with Dru Lavigne

LUCA FERRARI

Luca Ferrari: Dear Dru, first of all, it is a pleasure to have you on this magazine. You are a well know BSD-Guru, but could you please introduce yourself to our readers?

Dru Lavigne: I'm a Canadian citizen currently living in the Midwest of the US. I've been using, documenting and teaching BSD operating systems since 1996. For my day job, I manage the end-user documentation for the PC-BSD, FreeNAS, and TrueNAS operating systems and the Lumina desktop. I'm also a volunteer Director at the FreeBSD Foundation, and Founder and Chair of the BSD Certification Group, and as I find time, I'm a doc committer for the FreeBSD Project. When I'm not on a computer, you'll find me trialing new vegetable varieties in my backyard garden.

Luca Ferrari: How did you get interested in the FreeBSD world. Why did you choose to stay with BSD instead of other Unix-like operating systems?

Dru Lavigne: In the mid 90s, I went back to school to learn network and system administration. As graduation grew near and I startedlooking for a work, I noticed that all the interesting jobs wanted Unix skills. Wanting to increase my skills, and not having any money, I did an Internet search for "Free Unix". The first hit was freebsd.org. I went to the website and started reading the Handbook and thought "I can do this". Since I only had access to one computer and wanted to ramp up my skills quickly, I printed out the installation and networking chapters of the Handbook. I replaced the current operating system with FreeBSD and forced myself to learn how to do everything I needed to do on that computer in FreeBSD. It was a painful (and scary) few weeks as I figured out how to transition the family's workflow to FreeBSD, but it was also exhilarating to learn that "yes, I can do this!".

Since then, I've had the opportunity to try out or administer the other BSDs, several Linux distros, SCO, and Solaris. I found that the layout, logic, and release engineering process of the BSDs makes the most sense to me and I'm happiest when on a BSD system.

Luca Ferrari: You seem to be deeply involved with FreeBSD, do you also use other BSD systems? If yes, can you please enumerate and briefly describe your experience with them?

Dru Lavigne: I tend to use PC-BSD for desktops, FreeBSD for servers, and OpenBSD for firewalls. As part of the ongoing maintenance of the BSD certification program, I keep up with the release notes for each of the BSDs and perform test installs of each new release as preparation for the semi-annual update to the BSDA Study Guide.

Luca Ferrari: What do you suggest to a BSD user to keep up-to-date with the whole community? Do you recommend attending conferences, reading papers, books, manuals, or something different?

Dru Lavigne: There are quite a few resources for keeping up with the BSD community. BSDNow.tv has a weekly podcast where they discuss BSD topics. Every episode has been archived and the website also has tutorials. BSD Talk (*http://bsdtalk.blogspot.com/*) has an archive of interviews of people who working and using BSD. Both allow you to get caught up on with what's going on and who is doing what, if you have time to watch or listen.

If you are lucky enough to have a BSD conference come to a location near you, I highly recommend that you attend it. It is amazing to meet the people behind the IRC handles and email addresses that you run across in the BSD community in person. While as a first-time attendee you may initially feel that you're "just a user" or that the content will be over your head. I have yet to meet an attendee that didn't feel welcome in BSD community. I was glad that they attended, and was already making plans to attend another conference. If you can't make it to a BSD conference, you can still access the technical content, as presentations are archived at Youtube (search for the name of the conference).

Luca Ferrari: How many BSD-related conferences do you attend in a year?

Dru Lavigne: I attend a dozen or so open source and BSD conferences each year, where I help to staff the FreeBSD Foundation and PC-BSD or FreeNAS booth. I often present as well, on subjects such as ZFS and documentation. This year I attended the following conferences: FAST, SCALE, AsiaBSDCon, BSDCan, Essen DevSummit, vBSDCon, womENcourage, EuroBSDCon, LISA, and Fossetcon. I missed SELF, TLF, OLF, and the OpenZFS DevSummit this year due to scheduling conflicts.

Luca Ferrari: As an instructor, what is the most common difficulty people have in learning the Unix culture and, particularly, the BSD culture?

Dru Lavigne: Students who haven't been exposed to open source before are used to thinking of technology in terms of a purchasable brand consisting of "black boxes" that are supposed to "just work", without having to think about how they work. You can (and should) slow down and learn how things work. It can be a mind shift to learn that the freedom to use and change how something works does exist, and isn't considered stealing. And that learning how something works, while hard, can be fun.

Dru Lavigne

is a well known fellow in the BSD world. She has authored several technical books including BSD Hacks, The Best of FreeBSD Basics, and The Definitive Guide to PC-BSD. She is also a Director at the FreeBSD Foundation and Chair of the BSD Certification Group Inc.

BSD culture, in particular, is well suited for those who have the time and temperament to dive into how things work. With over 40 years of freely available source and commit messages, you can dive as deep as you want into learning how things came to be, how they evolved over the years, how they work now, and how they can be improved. There is a diverse range of stuff to choose from: from user tools to networking to memory management to hardware drivers to security mechanisms and so on. There is also a culture of sharing and learning and encouragement for users who demonstrate that they have done their homework and have their own ideas to contribute.

Luca Ferrari: Many universities, high schools, and middle schools are embracing Unix-like operating systems. This sounds good because students get in touch with the Unix culture and the Open Source world. What are your thoughts on this?

Dru Lavigne: I'm all for any program that encourages people to think, try things, figure things out, share ideas, and contribute their results. I'm also a big believer that people (of any age) should be introduced to a wide variety of technology, so that they can learn which technologies best meet their needs. This is especially important for students, so that they are aware that technology is more than just what happens to be in their classroom or what they see in advertisements.

Many schools also have programs or can partner with volunteer organizations that allow parents, older students, and technology professionals to assist in events, labs, and courses for introducing students to new technologies. I encourage readers to see what is available in their geographic area and to sign up for a volunteer event.

Luca Ferrari: What is the role of the FreeBSD Foundation and how can users contribute to the growth of the FreeBSD operating system through the Foundation?

Dru Lavigne: The FreeBSD Foundation is a US-based 501(c)3 non-profit organization, which supports and builds the FreeBSD Project and community worldwide. The Foundation receives donations from companies and individuals. Those donations are used to help provide hardware and infrastructure support for the FreeBSD Project, travel grants for FreeBSD contributors to attend conferences and DevSummits, project management and developer funding for specific development projects, and legal support for the Project and its logo, trademarks, and copyrights. As an individual, there are several ways you can help the Foundation support the FreeBSD Project. You can make a personal donation through the Foundation website, see if your employer has a donation matching program, or ask your employer if it is interested in making a corporate donation. If your business or employer uses FreeBSD in their product or infrastructure, contact the Foundation for assistance in writing a testimonial or case study on why you use FreeBSD. Let the Foundation know if you are doing interesting things with FreeBSD and would like to write an article for the FreeBSD Journal—they can assist you through the editorial process. If you are speaking about FreeBSD or staffing a FreeBSD booth at a local event or teaching a FreeBSD class, let the Foundation know so that they can help you spread the word. More information about the Foundation's activities, how to donate, and how to contact the Foundation can be found at *freebsdfoundation.org*.

Luca Ferrari: The BSD Certification Group aims at providing two levels of BSD certification, affordable to anyone. Could you please provide our readers some information about these certifications, and why it is important to become certified?

Dru Lavigne: The BSDA is an entry-level certification for BSD system administrators. It is available in English and Japanese as a multiple-choice exam which can be taken either at an exam event, such as a BSD conference, or at a testing center. The BSDP is an advanced certification for more experienced BSD system administrators. It is still under development and will be a 2-part exam with a lab component as well as a multiple-choice exam. More information about each exam is available at *http://www.bsdcertification.org/certification/*.

Both exams undergo a rigorous, psychometric assessment in order to provide value to both certificants and the employers who hire them. The exams are based on tasks that were deemed important by employers and the Certification Requirements document for each exam details the tasks covered by that exam. Achieving a certification assists system administrators in achieving a common base of knowledge. It can be used to fill skill gaps and to gauge which activities are important to employers. It also helps to grow the BSD community and demonstrates Certification that BSD is a viable enterprise solution and that skilled administrators are available for administering BSD systems.

Luca Ferrari: PC-BSD is a young project that is gaining more attention. What is your role in this project and why would you suggest to new users to try this operating system?

Dru Lavigne: I'm the editor and lead writer of the PC-BSD User Guide, which is published with each version of PC-BSD. I also assist in testing new features and beta versions, reporting bugs as I find them. Recently, I started the Lumina Handbook for the Lumina desktop which was created as part of the PC-BSD Project.

I've been using PC-BSD as my desktop since Kris Moore announced the Project in 2005. At that time, it provided a graphical installer and an alternate method for installing FreeBSD applications. Over the years, a Control Panel of graphical utilities was created for performing administrative tasks on the system—this was needed as most open source administrative tools were designed for Linux and did not understand FreeBSD device names, user management, firewall management, and so on. Once the tools were caught up, the Project started working on some interesting things, many of which take advantage of the ZFS filesystem. For example, with Life Preserver it is trivial to schedule automated ZFS snapshots (think backup of your files) and to timeslide between previous versions of files in order to restore a deleted or modified version of a file. These snapshots can also be scheduled to be replicated (backed up) to another system. If my main system blows up or I want to recreate it, I simply boot a PC-BSD installation disk on another computer, connect to the backup system, and restore the selected backup. Other cool tools include PersonaCrypt for logging into an encrypted USB stick (so that I can securely travel with my important files), switching to Tor mode for anonymous browsing, and safely updating the operating system or packages without fear that the update will break existing software. Should an upgrade be problematic, I simply reboot and select to boot into the version of the operating system before the update occurred. The backup, restore, and safe updates are features that I find necessary for my own desktop and which I miss when I'm not on a PC-BSD system. In this day and age, why aren't all filesystems and operating systems doing this?

Luca Ferrari: FreeNAS is another FreeBSD-based project which shares some PC-BSD technology, such as PBIs for example. What is your experience with this community and how does it compare to other FreeBSD-based project communities like, PC-BSD?

Dru Lavigne: This is a very large community and many of its users are new-comers to FreeBSD-based technologies, ZFS, and documentation that is updated and published with each release. There tends to be a learning curve for users to realize that the benefits of ZFS, particularly for storage, far outweigh the hardware requirements and that one should refer to the documentation first, rather than scouring the Internet for outdated, and often incorrect, howtos. The community seems to be mostly comprised of an even mix of Windows users and Linux users.

PC-BSD users traditionally tended to be Windows users looking for an alternative that was free, secure, and virus-free. We are now starting to see a lot of long-time Linux users who are looking for an alternative to systemd and who are curious about ZFS.

Luca Ferrari: What do you believe the BSD is still missing with regard to other operating systems?

Dru Lavigne: Visibility. The BSDs have a lot of cool, thoughtfully designed and well-implemented features that are either missing or poorly done in other operating systems. Some of today's buzzwords are based on technologies that have been available on BSDs for years. The perception is that functionalities are not available in the BSDs because they are called and implemented differently.

Luca Ferrari: What other projects are you currently involved in?

Dru Lavigne: I currently don't have the time to contribute to other projects, but there are other projects that I use. I recently converted the documentation I'm responsible for from wikis and OpenOffice to Sphinx (*sphinx-doc.org*). We also use other open source components in our doc toolchain, such as git for repository control, Jenkins (*jenkins-ci.org*) for continuous build integration, and Pootle (*pootle.translatehouse.org*) for managing both UI and documentation translations.

Luca Ferrari: In the endless battle amongst the text editors, what do you choose: Emacs or Vi?

Dru Lavigne: Definitely nvi (not vim).

ABOUT AUTHOR

Luca Ferrari lives in Italy with his wife and son. He received a PhD in Computer Science by University of Modena and Reggio Emilia, has been co-founder, member of the board of directors and president of Italian PostgreSQL Users' Group (ITPUG). Luca loves Open Source software and Unix culture, uses GNU Emacs, Perl, zsh and FreeBSD along with a lot of other cool tools.

Interview with Oleksandr Rybalko

LUCA FERRARI

The Raspberry PI Platform and The Challenges of Developing FreeBSD

Luca Ferrari: Dear Oleksandr, could you please introduce yourself to the BSDMag readers?
Oleksandr Rybalko: I'm a DevOps from Ukraine.
A long time ago, somewhere in the year 2000, I began to work as a support engineer in a small ISP in Mykolaiv city, Ukraine. The admin of that ISP gave us (support engineers) a free server to play with FreeBSD. That's the way I became a FreeBSD fan. Some time later, if I remember correctly, it was 2009, another good guy asked me to work on a FreeBSD embedded project. We selected the name ZRouter.org for it, because the main target was to run FreeBSD in small ARM/MIPS based routers with onboard Z-Wave transceiver. After some success of the project and several patches, Adrian Chadd asked me if I wished to become FreeBSD committer. Of course I agreed :)

Luca Ferrari: What is your involvement within the FreeBSD project?
Oleksandr Rybalko: I did three projects:

1. MIPS System-on-Chip Ralink RT3052 support;
2. ARM System-on-Chip Freescale i.MX515 basic support;
3. vt(4) virtual terminal.

So, from time to time I try to find spare minutes to support that code. And continue to improve the ZRouter.org project too.

Luca Ferrari: Are you currently working (or have worked) within other BSD projects?
Oleksandr Rybalko: Yes, it is ZRouter.org.

Luca Ferrari: Could you please introduce our readers to the Raspberry PI platform and detail what are the challenges of developing FreeBSD for such an embedded platform? What is the current status of the porting effort?
Oleksandr Rybalko: I think, my main help here was a USB OTG driver, which I wrote before for another device (Ralink RT3052), then port it to R-Pi. But it was rewritten by Hans Peter Selasky. I do not know so much about USB as Hans knows.

Another useful part of my help is Xorg support. I did a simple Xorg video driver which uses framebuffer exported by virtual terminal subsystem.

That is help to many guys to start use RPi as a simple desktop system.

Luca Ferrari: What are advantages of using FreeBSD on Raspberry PI with regard to other operating systems?
Oleksandr Rybalko: FreeBSD is very powerful as a network server. All modern network features in one box, with very fast processing.

Another good side of FreeBSD is modularity. It is not required to write code to use some driver that was already written for another system, you can just define it in configuration files (kernel config, kernel hints, FDT).

So if you want build a nice, R-Pi based, home server – use FreeBSD.

If you want to play with devices attached to R-Pi's GPIO – use FreeBSD.

Luca Ferrari: FreeBSD has a long history about embedded device installations, like for instance Soekris boards, did the experience and code produced along this path help the porting to the Raspberry PI?

Oleksandr Rybalko: Yes, for a long time I see “device migration” – when one device component is used in several different System-on-Chip(SoC)s, sometimes it’s very different, sometime it’s even in SoC made by another vendor. Plus, there are so called IP-cores. IP-core is just design of device which can be translated into device logic. So chip manufacturers get licenses to use some of it, and combine them into their own SoC design. As a result, we can find well known devices in very unexpected places. For example one USB On-The-Go controller, with a small difference, can be found in:

- Cavium Octeon MIPS64 SoC
- Ralink RT3052F MIPS32 SoC
- Broadcom BCM2835 ARM SoC (Raspberry Pi)

Luca Ferrari: What are the most notable use cases of FreeBSD running on Raspberry PI?

Oleksandr Rybalko: I don’t know :)

Some guys use RPi as a development board, because there are a lot of available pins to connect external devices. Some guys use it as lite PC.

I heard of one idea to show slides for conferences, but I don’t know if it was realized.

Luca Ferrari: What is the ZRouter project and what is the aim and challenges it is facing?

Oleksandr Rybalko: ZRouter.org project aims to embed FreeBSD into routers with very limited resources, like CPU clock speed, RAM size and Flash storage space. But there are a lot of systems which do not require full FreeBSD power, so ZRouter.org can be used there too.

For example, firewalls usually have a relatively big amount of resources, but it’s still an embedded machine, so does not require a full set of kernel modules and big number of utilities and libraries.

ZRouter.org can help there to build a system image with a reduced amount of modules/utils/libraries.

Nobody will write and compile programs at his firewall. :)

The main challenges of the ZRouter.org project were the big sizes of most software parts and lack of compressed, Flash oriented R/W filesystem. Those problems we worked around using some tricks, like use R/O filesystem compressed with LZMA (geom_uncompressed) + changes stored in tar.gz in a special flash partition.

Oleksandr Rybalko

is a DevOps with 15 years of experience as a developer (embedded/kernel/apps/web), a networking consultant, a system administrator, and a support specialist. With love to CADs and hardware design. Currently working as Consultant, Developer in D-Link Uk

Luca Ferrari: What would you suggest to a developer that wishes to deploy his own application/stack on Raspberry PI using FreeBSD?

Oleksandr Rybalko: Read wiki.FreeBSD.org and start working :)

There is nothing hard, even for a novice.

I have to have some love to FreeBSD, some love to your board, some wish to create new things and you will get what you like to see.

Anyway, the big and warm FreeBSD community will always help you to resolve your problem, but only if you're doing something. :)

And always remember "If you can get 90 percent of the desired effect for 10 percent of the work, use the simpler solution", meaning – always try to find the thing you want to implement before you start, maybe somebody did it before you. That also brings you closer to that thing FreeBSD developers very love – "code reuse"

Luca Ferrari: In the endless battle for the best editor in the world, could you please tell us which one do you use and what tools make your favourite toolchain?

Oleksandr Rybalko: Heh, you will not believe :)
I'm use mcedit, sometime vi/vim.

About toolchain
I like clang/llvm, but I hate to wait so long while it will be built during buildworld. And hate more to wait double build for embedded buildworld :)

ABOUT AUTHOR

Luca Ferrari lives in Italy with his wife and son. He received a PhD in Computer Science by University of Modena and Reggio Emilia, has been co-founder, member of the board of directors and president of Italian PostgreSQL Users' Group (ITPUG). Luca loves Open Source software and Unix culture, uses GNU Emacs, Perl, zsh and FreeBSD along with a lot of other cool tools.

NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- **CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)**
- **REMOTE MONITORING AND MANAGEMENT 24/7**
- **NETWORKING AND SECURITY (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)**
- **OS AND APPLICATION MANAGEMENT (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)**
- **DATABASE MANAGEMENT (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)**
- **MANAGED HOSTING IN CARRIER CLASS DATA CENTERS**
- **DISASTER RECOVERY**

Net Open SERVICES

WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS. **NETOPENSERVICES** TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS **15 YEARS OF EXPERIENCE** IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION. **NETOPENSERVICES** DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

WWW.NETOPENSERVICES.COM • CONTACT@NETOPENSERVICES.COM

Python Passive Network Mapping P2NMAP

CHET HOSMER TECHNICAL EDITOR GARY C. KESSLER

"It is by doubting that we come to investigate, and by investigating that we recognize the truth."

Peter Abelard

When performing incident response activities, mapping a network or performing penetration testing, you are likely to run in to situations where packet captures have already occurred. This could be in response to an event, or in today's world, more often as a routine practice. Either way, the packet capture (pcap) files can provide valuable information that we can examine and report on using P2NMAP-Analzer.py, which was developed in Chapter 4.

In order to accomplish this, I needed to develop a script that would extract the pertinent data from an existing pcap file and create both an .ipDict and .osDict file that can be processed. In other words, we need to interpret the pcap file to generate the same output files that P2NMAP-Capture.py does.

A number of years ago, Dug Song produced the Python Module `dpkt` (among many others) that is ideally suited for processing existing packet captures such as pcap files. I have tested the module extensively, and it is a nice addition to your core library within Python. One criticism of the library is the lack of documentation, however our use of the library is pretty straight-forward and my script will hopefully clear up the usage for at least our use case.

Installing `dpkt` as with most 3rd party Python packages is quite simple: The following command lines work just fine on Windows, Linux and Mac.

Windows

```
pip install dpkt
```

Linux/Mac

```
sudo pip install dpkt
```

Whenever I install a new package/module within Python, I run a quick verification that it is working. To do this, I can launch a Python shell from either the Windows or Linux command prompt. Below I show this from a Windows session. I then use the built-in Python `import` command to load the package. Once the package has been successfully imported you can then use the built-in Python `dir()` function to print the attributes associated with the package. For even more information you can also use the built-in `help()` function.

Note, if the import functions fails, it would indicate that the `dpkt` package is not properly installed (Code 1).

Review of P2NMAP-Capture

As you know from the development of the P2NMAP-Capture.py script for network mapping and OS Fingerprinting, we only require a few key pieces of data. We organize that data within an efficient data structure that both minimizes the size and also allows fast processing of the resulting data.

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Chet>python
Python 2.7.7 (default, Jun  1 2014, 14:17:13) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import dpkt
>>> dir(dpkt)
['Error', 'NeedData', 'PackError', 'Packet', 'UnpackError', '__author__', '__builtins__',
'__copyright__', '__doc__', '__file__', '__license__', '__name__', '__package__', '__path__'
, '__url__', '__version__', 'ah', 'aim', 'aoe', 'aoeata', 'aoecfg', 'arp', 'array', 'asn1', 'bgp',
'cdp', 'copy', 'crc32c', 'dhcp', 'diameter', 'dns', 'dpkt', 'dtp', 'esp', 'ethernet
', 'gre', 'gzip', 'h225', 'hexdump', 'hsrp', 'http', 'icmp', 'icmp6', 'ieee80211', 'igmp',
'in_cksum', 'in_cksum_add', 'in_cksum_done', 'ip', 'ip6', 'ipx', 'itertools', 'llc', 'loopb
ack', 'mrt', 'netbios', 'netflow', 'ntp', 'ospf', 'pcap', 'pim', 'pmap', 'ppp', 'pppoe', 'qq',
'radiotap', 'radius', 'rfb', 'rip', 'rpc', 'rtp', 'rx', 'sccp', 'sctp', 'sip', 'sll', '
smb', 'socket', 'ssl', 'ssl_ciphersuites', 'stp', 'struct', 'stun', 'tcp', 'telnet', 'tftp',
'tns', 'tpkt', 'udp', 'vrrp', 'yahoo']

>>> help(dpkt)
Help on package dpkt:

NAME
    dpkt - fast, simple packet creation and parsing.

FILE
    c:\python27\lib\site-packages\dpkt\__init__.py

PACKAGE CONTENTS
    ah
    aim
    aoe
    aoeata
    aoecfg
    arp
    asn1
    bgp
    cdp
    crc32c
    dhcp
    diameter
    dns
    dpkt
    dtp
    esp
    ethernet
    gre
    gzip
    h225
    hsrp
    http
    icmp
    icmp6
    ieee80211
    igmp
    ip
    ip6
    ipx
    llc
    loopback
    mrt
    netbios
    netflow
    ntp
    ospf
    pcap
    pim
    pmap
    ppp
    pppoe
    qq
    radiotap
    radius
    rfb
    rip
    rpc
    rtp
    rx
    sccp
    sctp
    sip
    sll
    smb
    snoop
    ssl
    ssl_ciphersuites
    stp
    stun
    tcp
    telnet
    tftp
    tns
    tpkt
    udp
    vrrp
    yahoo

DATA
    __author__ = 'Dug Song <dugsong@monkey.org>'
    __copyright__ = 'Copyright (c) 2004 Dug Song'
    __license__ = 'BSD'
    __url__ = 'http://dpkt.googlecode.com/'
    __version__ = '1.8.6'

VERSION
    1.8.6

AUTHOR
    Dug Song dugsong@monkey.org
```

```
import dpkt                          # 3rd Party Packet Parsing Module
from dpkt.udp import UDP             # Import specific objects from DPKT for convience
from dpkt.tcp import TCP             #

# Use dpkt and setup a pcapReader Object

pcapReader = dpkt.pcap.Reader(file(inFile, "rb"))

# Using the pcapReader Object process the
# the contents of the selected pcap file

# each interation through the loop will return
# 1) packet timestamp
# 2) packet raw data

for timeStamp, pckData in pcapReader:

    # Next I retrieve the etherNet packet contents


    etherNet = dpkt.ethernet.Ethernet(pckData)

    # Verify that this ethernet packet carries an IP Packet

    if etherNet.type == dpkt.ethernet.ETH_TYPE_IP:

        # get the ip data and extract the source and destination ip addresses
        # use the socket module to convert them to dot notational form

        # Decode the source and destination IP Address

        ip = etherNet.data
        sourceAddress      = socket.inet_ntoa(ip.src)
        destinationAddress = socket.inet_ntoa(ip.dst)
        # Check Packet Type (either TCP or UDP and process accordingly)

        if type(ip.data) == TCP :

            # Extract and Decode the Ports in use
            tcp = ip.data

            # Obtain Data for OS Fingerprinting

            # SYN Flag
            SYN = ( tcp.flags & dpkt.tcp.TH_SYN ) != 0

            # DF Flag
            DF = ( tcp.flags & dpkt.tcp.TH_URG ) != 0

            # Window Size
            WINDOW_SIZE = tcp.win

            # Time to Live and Type of Service values
            TTL = ip.ttl
            TOS = ip.tos

            # Now obtain the Source and Destination Port
            sourcePort      = tcp.sport
            destinationPort = tcp.dport
```

The core data we need from the pcap records in order to properly generate .ipDict and .osDict files are as follows:

General:

- Packet Timestamp
- .ipdict
- Source IP
- Destination IP
- Source Port
- Destination Port
- Protocol (TCP or UDP)
- .osDict
- Source IP
- Destination IP
- Source Port
- Destination Port
- SYN Flag
- DF Flag
- TTL (Time to live value)
- TOS (Type of service value)
- Window Size

Utilizing the dptk Package

The Code to extract the necessary data from the pcap files is isolated here (note: to simplify the code, I left out the exception processing, which is in the full version of the script). Minus the comment line, less than 20 lines of code are required to obtain the fields we require (Code 2).

The rest of the script uses the previously created classes:

```
class IPObservationDictionary:
class OSObservationDictionary:
```

.. along with the same packet processing code that was developed during the P2NMAP-Capture.py script. The full script is included here:

P2NMAP-PCAP-Extractor.py Script

See Code 3.

```
'''
Copyright (c) 2015 Chet Hosmer, cdh@python-forensics.org

Permission is hereby granted, free of charge, to any person obtaining a copy of this software
and associated documentation files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and permit persons to whom the Software is
furnished to do so, subject the following condition.
The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

'''
#
# Process .pcap files
# Create  .ipdict and .osDict result files
#         suitable for analysis with P2NMAP-Analyze
#                               and  P2NMAP-OS-Fingerprint

# import support functions

import argparse                 # Python Standard Library Parsing Module
import os                       # Python Standard Library OS module
import sys                      # Python Standard Library SYS Module
import socket                   # Python Standard Library socket module
import time                     # Python Standard Library time module
import datetime                 # Python Standard Library datetime module
import pickle                   # Python Standard Library pickling module

import dpkt                     # 3rd Party Packet Parsing Module
                                # pip install dptk    to intall the module
from dpkt.udp import UDP        # Import specific objects from DPKT for convience
from dpkt.tcp import TCP        #

# CONSTANTS

HOUR_INDEX = 3                  # Index of the Hour value in the Time Structure

#
# Name: ValDirWrite
#
# Desc: Function that will validate a directory path as
#       existing and writable.  Used for argument validation only
#
# Input: a directory path string
```

```
#
# Actions:
#               if valid will return the Directory String
#
#               if invalid it will raise an ArgumentTypeError within argparse
#               which will inturn be reported by argparse to the user
#

def ValDirWrite(theDir):

    # Validate the path is a directory
    if not os.path.isdir(theDir):
        raise argparse.ArgumentTypeError('Directory does not exist')

    # Validate the path is writable
    if os.access(theDir, os.W_OK):
        return theDir
    else:
        raise argparse.ArgumentTypeError('Directory is not writable')

#End ValDirWrite ===================================


#
# Name: ValidateFileRead Function
#
# Desc: Function that will validate that a file exists and is readable
#
# Input: A file name with full path
#
# Actions:
#               if valid will return path
#
#               if invalid it will raise an ArgumentTypeError within argparse
#               which will inturn be reported by argparse to the user
#

def ValFileRead(theFile):

    # Validate the path is a valid
    if not os.path.exists(theFile):
        raise argparse.ArgumentTypeError('File does not exist')

    # Validate the path is readable
    if os.access(theFile, os.R_OK):
        return theFile
    else:
        raise argparse.ArgumentTypeError('File is not readable')

#End ValidateFileRead ===================================



#
# Class: IPObservationDictionary
#
```

```
# Desc: Handles all methods and properties
#       relating to the IPOservations
#
#

class IPObservationDictionary:

    # Constructor

    def __init__(self):

        #Attributes of the Object

        self.Dictionary = {}             # Dictionary to Hold IP Observations

    # Method to Add an observation

    def AddOb(self, key, hour):

        # Check to see if key is already in the dictionary


        if key in self.Dictionary:

            # If yes, retrieve the current value
            curValue = self.Dictionary[key]

            # Increment the count for the current hour
            curValue[hour-1] = curValue[hour-1] + 1

            # Update the value associated with this key
            self.Dictionary[key] = curValue

        else:
            # if the key doesn't yet exist
            # Create one

            curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

            # Increment the count for the current hour
            curValue[hour-1] = curValue[hour-1] + 1

            self.Dictionary[key] = curValue

    # Print the Contents of the Dictionary

    def PrintOb(self):
        print "\nIP Observations"
        print "Unique Combinations:    ", str(len(self.Dictionary))
        print

        # Print Heading

        print '                                                ',
        print "|-------------------------------------------  Hourly Observations  --------------
-----------------------------------|"
        print '%16s' % "Server",
        print '%16s' % "Client",
        print '%7s'  % "Port",
        print '%5s'  % "Type",
        for i in range(0, 24):
            print  ' ',
            print '%02d' % i,
        print

        sorted = self.Dictionary.items()
        sorted.sort()

        # Print Contents
        for keys,values in sorted:

            print '%16s' % keys[0],
            print '%16s' % keys[1],
            print '%7s'  % str(keys[2]),
            print '%5s'  % keys[3],

            for i in range(0, 24):
                print '%4s' % str(values[i]),
            print

    # Save the Current Observation Dictionary
    # to the specified file

    def SaveOb(self, fileName):

        with open(fileName, 'wb') as fp:
            pickle.dump(self.Dictionary, fp)

    # Destructor Delete the Object

    def __del__(self):
        if VERBOSE:
            print "Closed"

# End IPObservationClass ====================================

#
# Class: OSObservationDictionary
#
# Desc: Handles all methods and properties
#       relating to the OSObservations
#
#

class OSObservationDictionary:

    # Constructor

    def __init__(self):

        #Attributes of the Object

        self.Dictionary = {}          # Dictionary to Hold IP Observations

    # Method to Add an observation

    def AddOb(self, key, hour):
        # Check to see if key is already in the dictionary

        if key in self.Dictionary:

            # If yes, retrieve the current value
            curValue = self.Dictionary[key]
            # Increment the count for the current hour
            curValue[hour-1] = curValue[hour-1] + 1

            # Update the value associated with this key
            self.Dictionary[key] = curValue

        else:
            # if the key doesn't yet exist
            # Create one

            curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

            # Increment the count for the current hour
            curValue[hour-1] = curValue[hour-1] + 1

            self.Dictionary[key] = curValue

    # Method to retrieve an observation
    # If no observation found return None

    def GetOb(self,key):
        if key in self.Dictionary:
            curValue = self.Dictionary[key]
            return curValue
        else:
            return None

    # Print the Contents of the Dictionary

    def PrintOb(self):

        print "\nOS Observations"
        print "Unique Combinations:    ", str(len(self.Dictionary))
        print


        # Print Heading
        print '                                           ',
        print "|-------------------------------------------  Hourly Observations  ---------------
------------------------------------|"

        print '%16s' % "Server",
        print '%4s'  % "TOS",
        print '%4s'  % "TTL",
        print '%6s'  % "DF",
        print '%7s'  % "Window",

        for i in range(0, 24):
            print ' ',
            print '%02d' % i,
        print "\n--------------------------------------------------------------------------------
----------------------------------------------------------------------------------"
```

```
        sorted = self.Dictionary.items()
        sorted.sort()

        # Print Contents
        for keys,values in sorted:
            print '%16s' % keys[0],
            print '%4s'  % str(keys[1]),
            print '%4s'  % str(keys[2]),
            print '%6s'  % str(keys[3]),
            print '%7s'  % str(keys[4]),

            for i in range(0, 24):
                print '%4s' % str(values[i]),
            print

    # End Print OS Observations
    # Save the Current Observation Dictionary
    # to the specified file

    def SaveOb(self, fileName):

        with open(fileName, 'wb') as fp:
            pickle.dump(self.Dictionary, fp)

    # Destructor Delete the Object

    def __del__(self):
        if VERBOSE:
            print "Closed"

# End OSObservationClass ====================================

#====================================
#
# Main Program Starts Here
#====================================

if __name__ == '__main__':

    # Setup Argument Parser Object

    parser = argparse.ArgumentParser('P2NAMP PCAP Extractor')

    parser.add_argument('-v', '--verbose', help="Provide Progress Messages", action='store_true')
    parser.add_argument('-o', '--outPath', type= ValDirWrite, required=True, help="Output
Directory")
    parser.add_argument('-i', '--inFile' , type= ValFileRead, required=True, help="PCAP input
File - Full Path")

    #process the command arguments

    cmdArgs = parser.parse_args()

    # convert arguments to simple local variables

    VERBOSE    = cmdArgs.verbose
    inFile     = cmdArgs.inFile
    outPath    = cmdArgs.outPath

    if VERBOSE:
        print "Packet Parsing Algorithm, version 1.0"
        print

        print "Opening Capture File: "+ inFile
        print
```

```
# Create IP observation dictionary object
ipOB = IPObservationDictionary()
osOB = OSObservationDictionary()

# Loop through all the packets found in the pcap file
# Obtain the timestamp and packet data

if VERBOSE:
    print "Processing PCAP, please wait ...\n"

# Use dpkt and setup a pcapReader Object
try:
    # Create pcapReader Object
    pcapReader = dpkt.pcap.Reader(file(inFile, "rb"))
except:
    # Error Reading pcap
    print "Error importing: ", infile
    quit()
# Using the pcapReader Object process the
# the contents of the selected pcap file
# each interation through the loop will return
# 1) packet timestamp
# 2) packet raw data

for timeStamp, pckData in pcapReader:

    # Next I retrieve the etherNet packet contents
    # and verify that it is an ethernet packet

    etherNet = dpkt.ethernet.Ethernet(pckData)

    # Verify that this ethernet packet carries an IP Packet

    if etherNet.type == dpkt.ethernet.ETH_TYPE_IP:

        # get the ip data and extract the source and destination ip addresses
        # use the socket module to convert them to dot notational form

        # Decode the source and destination IP Address
        ip = etherNet.data
        sourceAddress      = socket.inet_ntoa(ip.src)
        destinationAddress = socket.inet_ntoa(ip.dst)

        # Check Packet Type (either TCP or UDP and process accordingly)

        if type(ip.data) == TCP :

            # Extract and Decode the Ports in use
            tcp = ip.data

            # Obtain Data for OS Fingerprinting

            # SYN Flag
            SYN = ( tcp.flags & dpkt.tcp.TH_SYN ) != 0

            # DF Flag
            DF = ( tcp.flags & dpkt.tcp.TH_URG ) != 0

            # Window Size
            WINDOW_SIZE = tcp.win
        # Time to Live and Type of Service values
        TTL = ip.ttl
        TOS = ip.tos

        # Now obtain the Source and Destination Port
        sourcePort      = tcp.sport
        destinationPort = tcp.dport

        if sourcePort <= 1024:              # Assume server IP is server
            serverIP   = sourceAddress
            clientIP   = destinationAddress
            serverPort = sourcePort
            status = True
        elif destinationPort <= 1024:       # Assume destination IP is server
            serverIP   = destinationAddress
            clientIP   = sourceAddress
            serverPort = destinationPort
            status = True
        elif sourcePort <= destinationPort: # Assume server IP is server
            serverIP   = sourceAddress
            clientIP   = destinationAddress
            serverPort = sourcePort
            status = True
        elif sourcePort > destinationPort:  # Assume distinatin IP is server
            serverIP   = destinationAddress
            clientIP   = sourceAddress
            serverPort = destinationPort
            status = True
        else:                               # Should never get here
            serverIP   = "FILTER"
            clientIP   = "FILTER"
            serverPort = "FILTER"
            status = False

        # Convert the timestamp (epoch value)
        # into a time structure
        timeStruct = time.gmtime(timeStamp)

        # extract the hour the packet was captured
        theHour = timeStruct[HOUR_INDEX]

        if status:
            # Add a new IP observation and the hour
            ipOB.AddOb((serverIP, clientIP, serverPort, "TCP"), theHour)

            # If SYN is set also add a new OS Observation
            if SYN:
                osOB.AddOb( (serverIP, TOS, TTL, DF, WINDOW_SIZE), theHour)


    elif type(ip.data) == UDP :

        # Extract and Decode the Ports in use
        udp = ip.data
        sourcePort      = udp.sport
        destinationPort = udp.dport

        if sourcePort <= 1024:              # Assume server IP is server
```

```
                serverIP   = sourceAddress
                clientIP   = destinationAddress
                serverPort = sourcePort
                status = True
            elif destinationPort <= 1024:       # Assume destination IP is server
                serverIP   = destinationAddress
                clientIP   = sourceAddress
                serverPort = destinationPort
                status = True
            elif sourcePort <= destinationPort: # Assume server IP is server
                serverIP   = sourceAddress
                clientIP   = destinationAddress
                serverPort = sourcePort
                status = True
            elif sourcePort > destinationPort:  # Assume distinatin IP is server
                serverIP   = destinationAddress
                clientIP   = sourceAddress
                serverPort = destinationPort
                status = True
            else:                               # Should never get here
                serverIP   = "FILTER"
                clientIP   = "FILTER"
                serverPort = "FILTER"
                status = False

            # Convert the timestamp (epoch value)
            # into a time structure

            timeStruct = time.gmtime(timeStamp)
            theHour = timeStruct[3]

            if status:
                # Add a new observation and the hour
                ipOB.AddOb((serverIP, clientIP, serverPort, "UDP"), theHour)
        else:
            # Skip the Packet NOT TCP or UDP
            continue
    else:
        # skip this packet NOT Ethernet Type
        continue

# Once all packets are processed
# Print out Results

if VERBOSE:

    ipOB.PrintOb()
    osOB.PrintOb()

    print "\nSaving Observations ext: .ipDict and .osDict"

# Save observations in our compatible format

ipOutFile = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+".ipDict"
ipOutput  = os.path.join(outPath, ipOutFile)

osOutFile = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+".osDict"
osOutput  = os.path.join(outPath, osOutFile)

ipOB.SaveOb(ipOutput)
osOB.SaveOb(osOutput)

print 'Processing Complete'
```

Executing P2NMAP-PCAP-Extractor

Executing the PCAP-Extractor is done from the command line (again, Windows command shell along with Linux / Mac Shells all operate the same, see Code 4).

Executing the script with the –h option only, provides the argument list. Only 3 arguments are available:

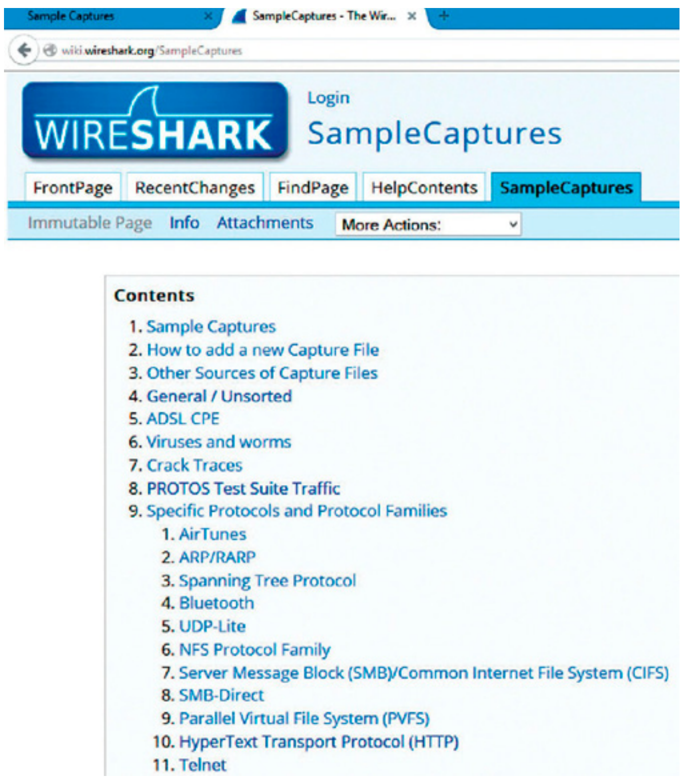
```
C:\CH5>python P2NMAP-PCAP-Extractor.py -h

usage: P2NAMP PCAP Extractor [-h] [-v] -o OUTPATH -i INFILE

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Provide Progress Messages
  -o OUTPATH, --outPath OUTPATH
                        Output Directory
  -i INFILE, --inFile INFILE
                        PCAP input File - Full Path
C:\CH5>python P2NMAP-PCAP-Extractor.py -v -i ./PCAP/test.pcap -o ./OUT/

Packet Parsing Algorithm, version 1.0

Opening Capture File: ./PCAP/test.pcap

Processing PCAP, please wait ...
```

- -v (optional) which will provide a verbose output from the application
- –i which is the input file and specifies the pcap file to extract from
- –o which specifies the output directory where the resulting .ipDict and. osDict files will be written with the familiar timestamp filename

```
Packet Parsing Algorithm, version 1.0

Opening Capture File: ./PCAP/test.pcap

Processing PCAP, please wait ...


IP Observations
Unique Combinations:     3984

                                                          |------------------------------------------
Hourly Observations  -------------------------------------------------------|
          Server              Client      Port  Type    00    01    02    03    04    05    06    07    08    09
10    11    12    13    14    15    16    17    18    19    20    21    22    23
         0.0.0.0  255.255.255.255          68   UDP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0     3     0     0
   107.20.103.220     172.16.133.26       443   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0   663     0     0
    107.20.158.52    172.16.133.153      8080   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0   708     0     0
   107.20.161.243     172.16.133.48       443   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0    57     0     0
    107.20.170.67     172.16.133.63       443   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0    40     0     0
   107.20.203.158    172.16.133.121        80   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0    20     0     0
   107.20.206.100     172.16.133.54        80   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0    36     0     0
    107.20.217.22     172.16.133.93        80   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0     6     0     0
   107.20.231.134     172.16.133.16        80   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0     9     0     0
   107.20.232.172    172.16.133.132        80   TCP     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0    24     0     0
```

When the script is executed with the verbose argument the following sample output is also generated on screen. (note: this output has been abridged to save space, See Code 5).

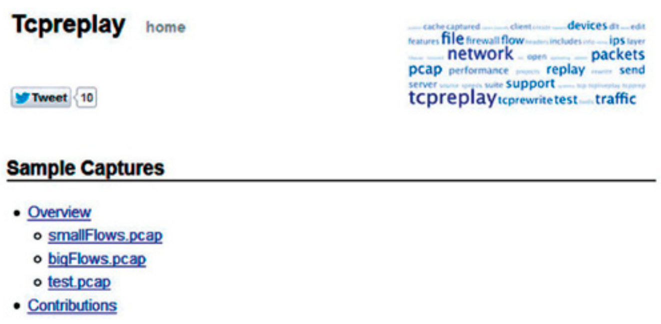
```
...
...  Output Abridged
...

OS Observations
Unique Combinations:     2477

                                                    |------------------------------------------  Hourly
Observations  ---------------------------------------------------|
          Server  TOS  TTL     DF  Window   00   01   02   03   04   05   06   07   08   09   10
11   12   13   14   15   16   17   18   19   20   21   22   23
------------------------------------------------------------------------------------------------
-----------------------------------------------------------------
  107.20.103.220    0  128  False    8192    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    2    0    0
  107.20.103.220   32   44  False    5840    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    2    0    0
  107.20.161.243    0  128  False    8192    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    3    0    0
  107.20.161.243   32   42  False    5840    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    3    0    0
   107.20.170.67    0  128  False    8192    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    2    0    0
   107.20.170.67   32   42  False   14600    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    2    0    0

Saving Observations ext: .ipDict and .osDict
Processing Complete
```



Contents

1. Sample Captures
2. How to add a new Capture File
3. Other Sources of Capture Files
4. General / Unsorted
5. ADSL CPE
6. Viruses and worms
7. Crack Traces
8. PROTOS Test Suite Traffic
9. Specific Protocols and Protocol Families
   1. AirTunes
   2. ARP/RARP
   3. Spanning Tree Protocol
   4. Bluetooth
   5. UDP-Lite
   6. NFS Protocol Family
   7. Server Message Block (SMB)/Common Internet File System (CIFS)
   8. SMB-Direct
   9. Parallel Virtual File System (PVFS)
   10. HyperText Transport Protocol (HTTP)
   11. Telnet

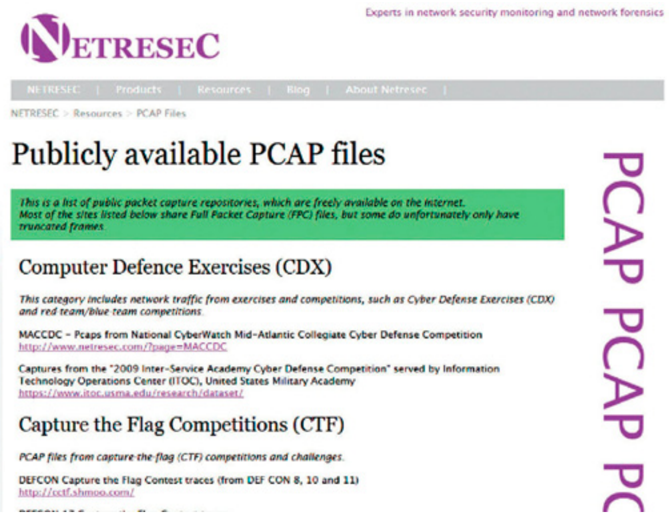
Figure 1. *Wireshark Samples Captures Web Page*


Tcpreplay
home
Tweet
10
cache captured client devices edit
features file firewall flow includes ips layer
network open packets
pcap performance replay send
server suite support
tcpreplay tcprewrite test traffic

Sample Captures

- Overview
  - smallFlows.pcap
  - bigFlows.pcap
  - test.pcap
- Contributions

Figure 2. *Tcpreplay Sample Captures*


Experts in network security monitoring and network forensics
NETRESEC
NETRESEC | Products | Resources | Blog | About Netresec |
NETRESEC > Resources > PCAP Files
PCAP PCAP PC

Publicly available PCAP files

This is a list of public packet capture repositories, which are freely available on the Internet. Most of the sites listed below share Full Packet Capture (FPC) files, but some do unfortunately only have truncated frames.

Computer Defence Exercises (CDX)

This category includes network traffic from exercises and competitions, such as Cyber Defense Exercises (CDX) and red team/blue team competitions.

MACCDC – Pcaps from National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition
http://www.netresec.com/?page=MACCDC

Captures from the "2009 Inter-Service Academy Cyber Defense Competition" served by Information Technology Operations Center (ITOC), United States Military Academy
https://www.itoc.usma.edu/research/dataset/

Capture the Flag Competitions (CTF)

PCAP files from capture-the-flag (CTF) competitions and challenges.

DEFCON Capture the Flag Contest traces (from DEF CON 8, 10 and 11)
http://cctf.shmoo.com/

Figure 3. *NETRESEC Sample Captures*

Now you can utilize the resulting files from this run:

```
20150303-151016.ipDict
20150303-151016.osDict
```

Now that we have generated the extracted ipDict and osDict files we can utilize P2NMAP-Analyze.py or P2NMAP-OS-Fingerprint.py to perform the requisite analysis. Note the P2NMAP-OS.Fingerpring.py script will be discussed in the next section.

Where do you find .pcap files to experiment with? You can obviously perform a Google search and you will find quite a few potential sources. However, three sources that I used heavily during experimentation include:

WireShark Samples Captures: *http://wiki.wireshark.org/SampleCaptures*

Tcpreplay: *http://tcpreplay.appneta.com/wiki/captures.html*

NETRESEC: *http://www.netresec.com/?page=PcapFiles*
Shown in Figure 1, Figure 2 and Figure 3

PASSIVE OS FINGERPRINTING

As many people are painfully aware, performing passive OS fingerprinting is a significant challenge. However, in this section I will provide the building blocks for identifying at least the general OS that is executing on the associated server platforms. The actual missing-link is a comprehensive dataset of rules that would more accurately map OS behaviors. This method is not meant to compete with the active methods of fingerprinting generated from the NMAP community, SAINT developers, McAfee/Intel Foundstone labs and other mainstream vendors. Rather the solution is presented to encourage expansion of the method, especially the painstaking task of developing signatures that will work during passive based examinations.

OS Fingerprinting Truth Table

During the passive collection of packet data (whether using P2NMAP-Capture. py or extracting packet data using P2NMAP-PCAP-Extractor.py) several key initial parameters were collected from observed packets. These IP packet values include Type of Service, Time to Live, Don't Fragment (DF) and Window Size. These values were only collected when the IP packet contained a TCP segment with the SYN flag set. This set of values allows for the creation of a Truth Table that would generate possible OS Fingerprinting when all four of the table values match the observed values.

Truth Tables provide a method of defining all possible values that can exist for a certain set of facts, variables or functions. These tables contains multiple rows and columns, with the top row representing the category values along with a final column that contains the conclusion based on the values specified in that row.

Table 1. shows a sample table with a few sample entries.

In order to improve on the basic concept, I wanted a bit more flexibility in the table. First, the *Time to Live* observations are impacted by the number of router hops that the packets take between the source and destination. Thus even if the packet starts out at 128, the value we observe is likely to be less than 128, therefore I will make this a range of values instead of a fixed number. Secondly, for certain known fingerprint signatures only 2 of the values may be required for an accurate identification.

For example, we may have knowledge that a certain CISCO network device has a starting TTL value of 255 and a *Window Size* of 4128, but the *Type of Service* and *DF* flags are not relevant, unknown or unreliable. In this case I would like to ignore the *DF* and *TOS* fields during the comparison (by using wild cards). Finally, for ease of parsing the table, the *TTL* and *Window Size* fields will always contain a range. The resulting truth table would then look like that in Table 2.

Table 1. *Basic Truth Table*

Time to Live	Type of Service	DF Flag	Window Size	OS Identified
128	0	T	5000-9000	Window NT
64	16	N	17520	Open BSD
128	0	T	32000-32768	Netware

Table 2. *Improved Truth Table*

Time to Live	Type of Service	DF Flag	Window Size	OS Identified
65-128	0	Y	5000-9000	Window NT
33-64	16	N	17520-17520	Open BSD
65-128	0	Y	32000-32768	Netware
129-255	*	*	4128-4128	Cisco IOS

A sample flat file truth table file is shown here, with the syntax being strict space delimited columns to make parsing the file simple. The file can be expanded to contain additional values as more known observations become available or the fingerprint data improves.

```
17-32    0   N 8192-8192     Windows
65-128   0   Y 5000-9000     Windows
65-128   0   Y 17000-18000   Windows
65-129   0   Y 32000-32768   Netware
33-64    16  N 17520-17520   OpenBSD
33-64    0   N 5804-5840     HP
33-64    0   N 24820-24280   SCO
33-64    16  Y 17520-17520   FreeBSD
33-64    16  N 17520 17520   Linux
33-64    0   Y 24280-24280   Solaris
65-128   0   N 65535-65535   CISCO
33-60    0   Y 16000-16100   AIX43
33-60    0   N 16000-16100   AIX43
65-128   0   Y 5000-9000     Windwos
127-255  192 N 3800-5000     Cisco
33-64    *   * 5720-5840     Linux
33-64    *   * 65535-65535   BSD
65-128   *   * 8192-8192     Windows
129-255  *   * 4128-4128     CISCO
```

Truth Table Python Class

To handle the processing of the truth table, I create a simple class that will perform three basic functions:

1. Load the truth table and process the range values
2. Accept a known set (TTL, TOS, DF and Window Size) as input and return the first matching OS Fingerprint from the loaded truth table.
3. Print the truth table for convenience and verification

```
class FingerPrint:

    # Constructor

    def __init__(self):

        self.classificationList = []

        self.osObservationsLoaded  = False
        self.osObservationFileName = ""
        self.osTruthTableLoaded = False
        self.osTruthTableFileName = ""
```

```
    # Load in the TruthTable from the userdefined text file

    def LoadTruthTable(self, truthTable):

        # String Index Values

        TTL_RANGE   = 0
        TOS         = 1
        DF          = 2
        WINDOW_SIZE = 3
        OS          = 4
        CHK_FLD     = 5
        tableErrors = False

        try:
            # Process the User Defined Input Table

            with open(truthTable, "r") as fileContents:

                for eachLine in fileContents:

                    values = eachLine.split()

                    # Make sure we have the proper number of fields
                    # in this line

                    if len(values) == CHK_FLD:
                        # convert the text ttl and winsize into low and high integers
                        # unless wild card specified
                        if values[TTL_RANGE] != '*':
                            ttlLow, ttlHigh = self.convertRange(values[TTL_RANGE])
                        else:
                            ttlLow = -1
                            ttlHgh = -1
                    if values[WINDOW_SIZE] != '*':
                        winLow, winHigh = self.convertRange(values[WINDOW_SIZE])
                    else:
                        winLow  = -1
                        winHigh = -1

                    #  Convert TOS to an integer, unless wild card
                    if values[TOS] == '*':
                        tosVal = '*'
                    else:
                        try:
                            tosVal = int(values[TOS])
                        except:
                            # invalid TOS value
                            # skip this line
                            tableErrors = True
                            continue

                    # Convert DF to True or False or wild card
                    if values[DF].upper()    == "Y":
                        dfVal = True
                    elif values[DF].upper() == "N":
                        dfVal = False
                    elif values[DF] == '*':
                        dfVal = "*"

                    else:
                        # invalid DF value
                        # skip this line
                        tableErrors = True
                        continue

                    if ttlLow != None and winLow != None:
                        self.classificationList.append( [ttlLow, ttlHigh, tosVal, dfVal,
winLow, winHigh, values[OS]] )
                    else:
                        tableErrors = True

            self.osTruthTableLoaded   = True
            self.osTruthTableFileName = truthTable

        except:
            print "***** Loading Truth Table - Failed *****"
            self.osTruthTableLoaded = False
            self.osTruthTableFileName = ""

        # Return to caller with errors flag
        # True  = Errors Found in Text File
        # False = All Rows Loaded Properly
        return tableErrors

    # End LoadTruthTable Method

    # Convert Range Method
    # Used to Convert range values 123-456
    # into two integer values

    def convertRange(self, theString):

        lowStr = ""
        hghStr = ""
    # parse the low value
    for x in range(0,len(theString)):
        if theString[x].isdigit():
            lowStr += theString[x]
        else:
            break
    # Skip the delimeters usually comma or dash
    for s in range(x, len(theString)):
        if theString[s].isdigit():
            break
        else:
            continue

    # If we are not at the end
    if s < len(theString):
        # parse the high value
        for y in range(s, len(theString)):
            if theString[y].isdigit():
                hghStr += theString[y]
            else:
                break
    else:
        return None, None

    # If we have two strings, then convert to ints

    if len(lowStr) > 0 and len(hghStr) > 0:
        lowVal = int(lowStr)
        hghVal = int(hghStr)
    else:
        return None, None

    # Finally,
    # If we have a proper low high relationship return the ints
    if lowVal <= hghVal:
        return lowVal, hghVal
    else:
        return None, None

# End convertRange Method


# GetOSClassification Searches the Loaded Truth Table
# for a match, it will return on the first successful match
# if no match is found it will return the string "UNDEFINED"

def GetOSClassification(self, ttl, tos, df, winSize):

    # List Index

    TTL_LOW     = 0
    TTL_HGH     = 1
    TOS_VAL     = 2
    DF_VAL      = 3
    WIN_LOW     = 4
    WIN_HGH     = 5
    OS_VAL      = 6
    # Search the classificationList (TruthTable)

    for entry in self.classificationList:
            # First Check the TTL Value, if in Range continue
            if ( (entry[TTL_LOW] <= ttl and entry[TTL_HGH] >= ttl) or (entry[TTL_LOW] == '*') ):
                # Next Check the Type of Service Value, if Match Continue
                if entry[TOS_VAL] == tos  or entry[TOS_VAL] == "*":
                    # Next Check the DF Flag, if Match Continue
                    if entry[DF_VAL] == df or entry[DF_VAL] == "*":
                        # Finally, check the Window Size, if in Range Continue
                        if ( (entry[WIN_LOW] <= winSize and entry[WIN_HGH] >= winSize) or
(entry[WIN_LOW] == '*') ):
                            # Return the OS Value Found
                            return entry[OS_VAL]

        # If none of the rules result in a match
        return "UNDEFINED"

    # End GetOSClassification Method


    # PrintTruthTable Method
    # Print out the currently loaded Truth Table


    def PrintTruthTable(self):

        TTL_LOW     = 0
        TTL_HGH     = 1
        TOS_VAL     = 2
        DF_VAL      = 3
        WIN_LOW     = 4
        WIN_HGH     = 5
        OS_VAL      = 6

        print >> OUT,'\nCurrent Loaded Fingerprint Truth Table\n'
        print >> OUT,'%10s ' % 'TTL RANGE',

        print >> OUT,'%4s ' % 'TOS',
        print >> OUT,'%5s ' % ' DF ',

        print >> OUT,'%16s ' % 'WIN RANGE',

        print >> OUT,'%22s ' % 'OS Fingerprint'
        print >> OUT,'================================================================='

        for entry in self.classificationList:

            print >> OUT,'%3s ' % str(entry[TTL_LOW]),
            print >> OUT, "-",
            print >> OUT,'%3s ' % str(entry[TTL_HGH]),

            print >> OUT,'%4s ' % str(entry[TOS_VAL]),
            print >> OUT,'%5s ' % str(entry[DF_VAL]),

            print >> OUT,'%6s ' % str(entry[WIN_LOW]),
            print >> OUT, "-",
            print >> OUT,'%6s ' % str(entry[WIN_HGH]),

            print >> OUT,' %20s '% entry[OS_VAL]

        print >> OUT, '\n\n'

    # End PrintTruthTable Method
```

Now that we can load and process the truth table, all that is left to do is build a menu driven script that can:

1. Load a previously generated .osDict file
2. Load and process and user defined truth table
3. Generate the OS fingerprint results

In addition, I have provided similar support functions as with the P2NMAPAnalyze script to allow directing the output to a file, along with the ability to print the contents of the .osDict observations and the truth table contents.

P2NMAP-OS-Fingerprint Script

```
'''
Copyright (c) 2015 Chet Hosmer, cdh@python-forensics.org

Permission is hereby granted, free of charge, to any person obtaining a copy of this software
and associated documentation files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and permit persons to whom the Software is
furnished to do so, subject the following condition.
The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

'''

#
# P2NMA-Analyze.py Script
#
# Perform analysis of previously capture .ipdict files
#
#
# Version 1.0 February 17-2015

import argparse          # Python Standard Library - Parser for command-line options, arguments
import os                # operating system functions i.e. file I/o
import datetime          # Python Standard Library date and time methods
import pickle            # Python Standard Library pickle methods
import socket            # Python Standard Library Low Level Networking
import sys               # Python Standard Library Low Level System Methods

# PSUEDO CONSTANTS

PRINT_STDOUT    = True   # If True, all output and menu selections directed
                         # If False, all output directed to a file except menu

OUT             = sys.stdout          # Default Output to Standard Out
#
# Class: OSObservationDictionary
#
# Desc: Handles all methods and properties
#       relating to the OSObservations
#
#

class OSObservationDictionary:

    # Constructor

    def __init__(self):

        #Attributes of the Object

        self.Dictionary = {}              # Dictionary to Hold IP Observations
        self.osObservationFileName = ""
        self.osObservationsLoaded  = False
    def LoadOb(self, fileName):
        try:
            with open(fileName, 'rb') as fp:
                self.Dictionary = pickle.loads(fp.read())
                self.observationFileName = fileName
                self.observationsLoaded  = True
        except:
            print "Loading OS Observations File - Failed"
            self.osObservationsLoaded  = False
            self.osObservationFileName = ""

        # Method to retrieve an observation
    # If no observation found return None

    def GetOb(self,key):

        if key in self.Dictionary:
            curValue = self.Dictionary[key]
            return curValue
        else:
            return None

    def GetAllObservations(self):

        observationList = []

        sorted = self.Dictionary.items()
        sorted.sort()

        for k, v in sorted:
            observationList.append(k)

        return observationList

    def PrintOb(self):

        print >> OUT, "\nOS Observations"
        print >> OUT, "Unique Combinations:    ", str(len(self.Dictionary))
        print >> OUT


        # Print Heading
        print >> OUT, '                                           ',
        print >> OUT, "|-------------------------------------------  Hourly Observations  -------
--------------------------------------------|"

        print >> OUT, '%16s' % "Server",
        print >> OUT, '%4s'  % "TOS",
        print >> OUT, '%4s'  % "TTL",
        print >> OUT, '%6s'  % "DF",
        print >> OUT, '%7s'  % "Window",

        for i in range(0, 24):
            print >> OUT,  ' ',
            print >> OUT, '%02d' % i,
        print >> OUT, "\n-------------------------------------------------------------------------
-------------------------------------------------------------------------------------------"

        sorted = self.Dictionary.items()
        sorted.sort()
        # Print Contents
        for keys,values in sorted:
            print >> OUT, '%16s' % keys[0],
            print >> OUT, '%4s'  % str(keys[1]),
            print >> OUT, '%4s'  % str(keys[2]),
            print >> OUT, '%6s'  % str(keys[3]),
            print >> OUT, '%7s'  % str(keys[4]),

            for i in range(0, 24):
                print >> OUT, '%4s' % str(values[i]),
            print >> OUT

    # End Print OS Observations

    # Load in and Observation Dictionary
    # from the specified file

    def LoadOb(self, fileName):

        try:
            with open(fileName, 'rb') as fp:
                self.Dictionary = pickle.loads(fp.read())
                self.osObservationFileName = fileName
                self.osObservationsLoaded  = True
        except:
            print "Loading Observations - Failed"
            self.osObservationsLoaded  = False
            self.osObservationFileName = ""


    # Destructor Delete the Object

    def __del__(self):
        print >> OUT, "OS Observation Dictionary Closed"

# End OSObservationClass ====================================

class FingerPrint:

    # Constructor

    def __init__(self):


        self.classificationList = []

        self.osObservationsLoaded  = False
        self.osObservationFileName = ""
        self.osTruthTableLoaded = False
        self.osTruthTableFileName = ""

    # Load in the TruthTable from the userdefined text file

    def LoadTruthTable(self, truthTable):

        # String Index Values

        TTL_RANGE   = 0
        TOS         = 1
        DF          = 2
        WINDOW_SIZE = 3
        OS            = 4
        CHK_FLD       = 5

        tableErrors = False

        try:
            # Process the User Defined Input Table

            with open(truthTable, "r") as fileContents:

                for eachLine in fileContents:

                    values = eachLine.split()

                    # Make sure we have the proper number of fields
                    # in this line

                    if len(values) == CHK_FLD:
                        # convert the text ttl and winsize into low and high integers
                        # unless wild card specified
                        if values[TTL_RANGE] != '*':
                            ttlLow, ttlHigh = self.convertRange(values[TTL_RANGE])
                        else:
                            ttlLow = -1
                            ttlHgh = -1

                        if values[WINDOW_SIZE] != '*':
                            winLow, winHigh = self.convertRange(values[WINDOW_SIZE])
                        else:
                            winLow  = -1
                            winHigh = -1

                        #  Convert TOS to an integer, unless wild card
                        if values[TOS] == '*':
                            tosVal = '*'
                        else:
                            try:
                                tosVal = int(values[TOS])
                            except:
                                # invalid TOS value
                                # skip this line
                                tableErrors = True
                                continue

                        # Convert DF to True or False or wild card
                        if values[DF].upper()   == "Y":
                            dfVal = True
                        elif values[DF].upper() == "N":
                            dfVal = False
                        elif values[DF] == '*':
                            dfVal = "*"
                        else:
                            # invalid DF value
                            # skip this line
                            tableErrors = True
                            continue

                        if ttlLow != None and winLow != None:
                            self.classificationList.append( [ttlLow, ttlHigh, tosVal, dfVal,
winLow, winHigh, values[OS]] )
                        else:
                            tableErrors = True
            self.osTruthTableLoaded   = True
            self.osTruthTableFileName = truthTable

        except:
            print "***** Loading Truth Table - Failed *****"
            self.osTruthTableLoaded = False
            self.osTruthTableFileName = ""

        # Return to caller with errors flag
        # True  = Errors Found in Text File
        # False = All Rows Loaded Properly

        return tableErrors

    # End LoadTruthTable Method

    # Convert Range Method
    # Used to Convert range values 123-456
    # into two integer values

    def convertRange(self, theString):

        lowStr = ""
        hghStr = ""

        # parse the low value
        for x in range(0,len(theString)):
            if theString[x].isdigit():
                lowStr += theString[x]
            else:
                break
        # Skip the delimeters usually comma or dash
        for s in range(x, len(theString)):
            if theString[s].isdigit():
                break
            else:
                continue

        # If we are not at the end
        if s < len(theString):
            # parse the high value
            for y in range(s, len(theString)):
                if theString[y].isdigit():
                    hghStr += theString[y]
                else:
                    break
        else:
            return None, None

        # If we have two strings, then convert to ints

        if len(lowStr) > 0 and len(hghStr) > 0:
            lowVal = int(lowStr)
            hghVal = int(hghStr)
        else:
            return None, None

        # Finally,
        # If we have a proper low high relationship return the ints
        if lowVal <= hghVal:
            return lowVal, hghVal
        else:
            return None, None

    # End convertRange Method
    # GetOSClassification Searches the Loaded Truth Table
    # for a match, it will return on the first successful match
    # if no match is found it will return the string "UNDEFINED"

    def GetOSClassification(self, ttl, tos, df, winSize):

        # List Index

        TTL_LOW     = 0
        TTL_HGH     = 1
        TOS_VAL     = 2
        DF_VAL      = 3
        WIN_LOW     = 4
        WIN_HGH     = 5
        OS_VAL      = 6

        # Search the classificationList (TruthTable)

        for entry in self.classificationList:

            # First Check the TTL Value, if in Range continue
            if ( (entry[TTL_LOW] <= ttl and entry[TTL_HGH] >= ttl) or (entry[TTL_LOW] == '*') ):
                # Next Check the Type of Service Value, if Match Continue
                if entry[TOS_VAL] == tos  or entry[TOS_VAL] == "*":
                    # Next Check the DF Flag, if Match Continue
                    if entry[DF_VAL] == df or entry[DF_VAL] == "*":
                        # Finally, check the Window Size, if in Range Continue
                        if ( (entry[WIN_LOW] <= winSize and entry[WIN_HGH] >= winSize) or
(entry[WIN_LOW] == '*') ):
                            # Return the OS Value Found
                            return entry[OS_VAL]

        # If none of the rules result in a match
        return "UNDEFINED"

    # End GetOSClassification Method


    # PrintTruthTable Method
    # Print out the currently loaded Truth Table

    def PrintTruthTable(self):

        TTL_LOW     = 0
        TTL_HGH     = 1
        TOS_VAL     = 2
        DF_VAL      = 3
        WIN_LOW     = 4
        WIN_HGH     = 5
        OS_VAL      = 6

        print >> OUT,'\nCurrent Loaded Fingerprint Truth Table\n'
        print >> OUT,'%10s ' % 'TTL RANGE',

        print >> OUT,'%4s ' % 'TOS',
        print >> OUT,'%5s ' % ' DF ',

        print >> OUT,'%16s ' % 'WIN RANGE',

        print >> OUT,'%22s ' % 'OS Fingerprint'
        print >> OUT,'==================================================================='
        for entry in self.classificationList:

            print >> OUT,'%3s ' % str(entry[TTL_LOW]),
            print >> OUT, "-",
            print >> OUT,'%3s ' % str(entry[TTL_HGH]),

            print >> OUT,'%4s ' % str(entry[TOS_VAL]),
            print >> OUT,'%5s ' % str(entry[DF_VAL]),

            print >> OUT,'%6s ' % str(entry[WIN_LOW]),
            print >> OUT, "-",
            print >> OUT,'%6s ' % str(entry[WIN_HGH]),

            print >> OUT,' %20s '% entry[OS_VAL]

        print >> OUT, '\n\n'

    # End PrintTruthTable Method

    # Print the OS Fingerprint Analysis Menu

    def PrintOSAnalysisMenu(self, osState, osFile):

        print "\n========== P2NMAP OS Fingerprint Analyze Menu ==========\n"

        if osState:
            print "Current Observation File: ", osFile
        if self.osTruthTableLoaded:
            print "Current OS Truth Table:   ", self.osTruthTableFileName

        print

        print "[L]    Load Observation File for Analysis"
        print "[T]    Load Observation Truth Table"

        if osState and self.osTruthTableLoaded:

            if PRINT_STDOUT:
                print "[O]    Direct Output to File   (Current = Stdout)"
            else:
                print "[O]    Direct Output to Stdout (Current = results.txt)"

            print "============================================================="
            print "[1]    Print Truth Table"
            print "[2]    Print Observations"
            print "[3]    Print Probable OS Fingerprint "
            print
        print "[X]    Exit P2NMAP Fingerprint Analysis"
        print

# End FingerPrint Class ====================================


#====================================
# Main Program Starts Here
#====================================
if __name__ == '__main__':

    # Local Psuedo Constants
    IP  = 0
    TOS = 1
    TTL = 2
    DF  = 3
    WIN = 4

    # Instantiate the FingerPrint and OSObservationDictionary Objects
    fpOB = FingerPrint()
    osOB = OSObservationDictionary()

    # Process User Input
    while True:
        fpOB.PrintOSAnalysisMenu(osOB.osObservationsLoaded, osOB.osObservationFileName)
        menuSelection = raw_input("Enter Selection: ").upper()

        # Attempt to Load the OS Capture File
        if menuSelection == 'L':
            fileName = raw_input("Enter OS Capture File: ")
            osOB.LoadOb(fileName)
            print

        # Attempt to Load a Truth Table
        elif menuSelection == 'T':
            fileName  = raw_input("Enter Truth Table File: ")
            rowErrors = fpOB.LoadTruthTable(fileName)

            # If Table Loaded, then check for row errors
            if fpOB.osTruthTableLoaded:
                # If rowError then inform the user
                if rowErrors:
                    print >> OUT, "Table Loaded but with Errors, Check Truth Table Input"
                else:
                    print >> OUT, "Truth Table Loaded"
                print

        # Toggle the Current Output State
        elif menuSelection == 'O':
            if PRINT_STDOUT:
                PRINT_STDOUT = False
                OUT = open("results.txt", 'w+')
            else:
                PRINT_STDOUT = True
                OUT.close()
                OUT = sys.stdout

        # Print Out the Current Truth Table

        elif menuSelection == '1':
            fpOB.PrintTruthTable()


        # Print out the Current Observation List
        elif menuSelection == '2':
            osOB.PrintOb()

        # Process the Observation List and
        # Print out Server Type
        # By matching the observed value list with
        # the loaded Truth Table
        elif menuSelection == '3':

            obList = osOB.GetAllObservations()

            print >> OUT
            print >> OUT,'%16s ' % 'IP Address',
            print >> OUT,'%25s ' % 'Fingerprint OS Type'
            print >> OUT,"============================================================"

            for entry in obList:
                osType = fpOB.GetOSClassification(entry[TTL], entry[TOS], entry[DF], entry[WIN])
                print >> OUT,'%16s ' % entry[IP],
                print >> OUT,'%25s ' % osType


        elif menuSelection == 'X':
            break
        else:
            print "Entry not recognized"
            continue

    OUT.flush()

print "done"
```

Executing P2NMAP-OS-Fingerprint

Operating P2NMAP-OS-Fingerprint.py, requires no command line arguments as the user is prompted for all necessary input.

```
C:\CH5>python P2NMAP-OS-FingerPrint.py

========== P2NMAP OS Fingerprint Analyze Menu ==========


[L]     Load Observation File for Analysis
[T]     Load Observation Truth Table
[X]     Exit P2NMAP Fingerprint Analysis

Enter Selection:
```

Before processing and generating the OS Fingerprints a valid observation file and a valid truth table must be provided: Once this is accomplished successfully, the menu will change to allow for the execution of the remaining options:

1. Print truth table
2. Print observations
3. Print probable OS fingerprint

```
========== P2NMAP OS Fingerprint Analyze Menu ==========

Current Observation File:  test1.osDict
Current OS Truth Table:    fptt.txt

[L]     Load Observation File for Analysis
[T]     Load Observation Truth Table
[O]     Direct Output to File     (Current = Stdout)
============================================================
[1]     Print Truth Table
[2]     Print Observations
[3]     Print Probable OS Fingerprint

[X]     Exit P2NMAP Fingerprint Analysis

Enter Selection: 1
```

Selecting Option 1, produces the following truth table output, (Current Loaded Fingerprint Truth Table).

```
TTL RANGE    TOS    DF          WIN RANGE           OS Fingerprint
=================================================================
 17  -  32     0  False     8192  -   8192             Windows
 65  - 128     0   True     5000  -   9000             Windows
 65  - 128     0   True    17000  -  18000             Windows
 65  - 129     0   True    32000  -  32768             Netware
 33  -  64    16  False    17520  -  17520             OpenBSD
 33  -  64     0  False     5804  -   5840                  HP
 33  -  64    16   True    17520  -  17520             FreeBSD
 33  -  64     0   True    24280  -  24280             Solaris
 65  - 128     0  False    65535  -  65535               CISCO
 33  -  60     0   True    16000  -  16100               AIX43
 33  -  60     0  False    16000  -  16100               AIX43
 65  - 128     0   True     5000  -   9000             Windwos
127  - 255   192  False     3800  -   5000               Cisco
 33  -  64     *      *     5720  -   5840               Linux
 33  -  64     *      *    65535  -  65535                 BSD
 65  - 128     *      *     8192  -   8192             Windows
129  - 255     *      *     4128  -   4128               CISCO
```

Selecting Option 2, produces the familiar (abridged) OS Observations result

```
========== P2NMAP OS Fingerprint Analyze Menu ==========

Current Observation File:  test1.osDict
Current OS Truth Table:    fptt.txt

[L]      Load Observation File for Analysis
[T]      Load Observation Truth Table
[O]      Direct Output to File     (Current = Stdout)
==============================================================
[1]      Print Truth Table
[2]      Print Observations
[3]      Print Probable OS Fingerprint

[X]      Exit P2NMAP Fingerprint Analysis

Enter Selection:  2
```

```
OS Observations
Unique Combinations:      2477

                                                    |--------------------------------------------------  Hourly
Observations  ---------------------------------------------------------|
            Server  TOS  TTL    DF  Window    00    01    02    03    04    05    06    07    08    09    10
11    12    13    14    15    16    17    18    19    20    21    22    23
-----------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------
  107.20.103.220     0  128  False    8192     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     2     0     0
  107.20.161.243     0  128  False    8192     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     3     0     0
    107.20.170.67    0  128  False    8192     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     2     0     0
    107.20.170.67   32   42  False   14600     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     2     0     0
```

Finally, Selecting Option 3, produces the (abridged) Probable OS Fingerprint Result

```
========== P2NMAP OS Fingerprint Analyze Menu ==========

Current Observation File:  test1.osDict
Current OS Truth Table:    fptt.txt

[L]      Load Observation File for Analysis
[T]      Load Observation Truth Table
[O]      Direct Output to File     (Current = Stdout)
==============================================================
[1]      Print Truth Table
[2]      Print Observations
[3]      Print Probable OS Fingerprint

[X]      Exit P2NMAP Fingerprint Analysis

Enter Selection:  3



      IP Address          Fingerprint OS Type
==============================================================
  107.20.103.220                             Linux
  107.20.161.243                             Linux
   107.20.170.67                           Windows
…
…
…

   98.139.51.132                               BSD
   98.142.99.171                             Linux
    99.61.13.155                           Windows
```

REVIEW

I tackled extracting key data from pcap files to convert them into the .ipDict and .osDict format in Chapter 5. This provides a direct way of handling captured network traffic from sources other than P2NMAP-Capture.py developed in Chapter 3. This was critical since more and more organizations are routinely collecting, preserving and retaining pcap files in their normal course of business. To extract the data, we used the 3rd Party Python Library `dpkt`, and were able to accomplish this core extraction process in less than 20 lines of code. I then wrapped this process into a script to automatically perform the functions.

Next, for the first time we used the contents of the .osDict file to make use of the observed TTL, TOS, DF and Window Size to predict the OS Type of the server in question. I defined a method using a truth table to perform this operation, and provided a baseline for further expansion of the truth table to improve the accuracy of the fingerprint identification. Next, I created the complete script, P2NMAP-OS-Fingerprint.py, to experiment with this new method of OS identification.

I also provided sample script execution for both P2NMAP-PCAP-Extractor and
P2NMAP-OS-Fingerprint.

SUMMARY QUESTIONS

1. Challenge Problem 1: Develop experiments that generate observed behavior of a variety of operating systems under normal operation. Use that data to improve the truth table and ultimately the accuracy of Passive OS Fingerprint identification.
2. Challenge Problem 2: Utilize the ipDict result and the port values obtained to further improve OS Fingerprint identification by creating a truth table that pro-

vides association of known ports with the most operating system most probably in use.

3. Challenge Problem 3: Modify both P2NMAP-Capture.py and P2NMAPPCAP-Extractor.py to collecting TTL, TOS, DF and Window Size observations for protocols other than TCP/UDP.

Additional Resources

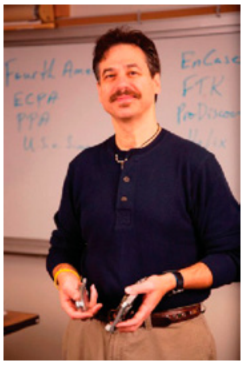
- Song Dug, dptk Python Package, *https://pypi.python.org/pypi/dpkt*
- Silverman Jeffery, dptk documentation, *http://www.commercialventvac.com/dpkt.html*

CHET HOSMER

is the Founder of Python Forensics, Inc. a non-profit organization focused on the collaborative development of open-source investigative technologies using the Python programming language. Chet is also the founder of WetStone Technologies, Inc. and has been researching and developing technology and training surrounding forensics, digital investigation and steganography for over two decades. He has made numerous appearances to discuss emerging cyber threats including National Public Radio's Kojo Nnamdi show, ABC's Primetime Thursday, NHK Japan, CrimeCrime TechTV and ABC News Australia. He has also been a frequent contributor to technical and news stories relating to cyber security and forensics and has been interviewed and quoted by IEEE, The New York Times, The Washington Post, Government Computer News, Salon.com and Wired Magazine.

Chet serves as a visiting professor at Utica College where he teaches in the Cybersecurity Graduate program. He is also an Adjunct Faculty member at Champlain College in the Masters of Science in Digital Forensic Science Program. Chet delivers keynote and plenary talks on various cyber security related topics around the world each year.
Chet resides with Wife Janet, Son Matthew along with his four legged family near Myrtle Beach, South Carolina.

GARY C. KESSLER, PH.D., CCE, CCFP, CISSP,



is a Professor of Homeland Security at Embry-Riddle Aeronautical University, a member of the North Florida ICAC (Volusia County Sheriff's Department), and president and janitor of Gary Kessler Associates, a training and consulting company specializing in computer and network security and digital forensics. He is the co-author of two professional texts and over 70 articles, a frequent speaker at regional, national, and international conferences, and past editor-in-chief of the Journal of Digital Forensics, Security and Law. More information about Gary can be found at his Web site, http://www.garykessler.net.
PCAP Extractor and OS Fingerprinting

Dr.WEB®
since 1992

The ERA of harmony and security

New Dr.Web! version 10

- Brand new user interface
- Configuration as simple as ABC
- Honest protection against real threats

Comprehensive protection for Windows
Anti-virus for Mac OS X and Linux

Basic protection for Windows,
Mac OS X and Linux

Protection for any and all devices*

Home and office

New purchase and renewal

+150

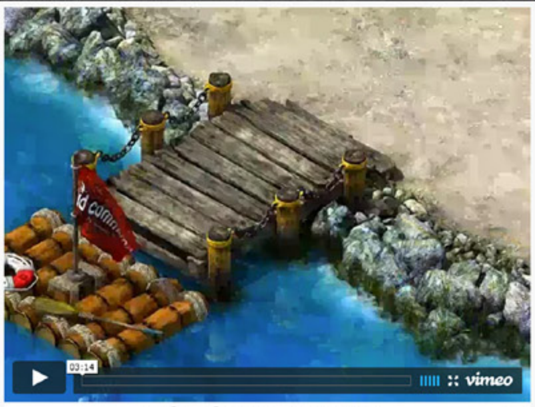
* PC, Mac and mobile devices running OS supported by Dr.Web.

Protection for mobile devices — for free!

Doctor Web is the Russian developer of Dr.Web anti-virus software. Dr.Web anti-virus software has been developed since 1992. Doctor Web is one of the few anti-virus vendors in the world to have its own technologies to detect and cure malware. Dr.Web anti-virus software allows IT environments to effectively withstand any threats, even those not yet known.

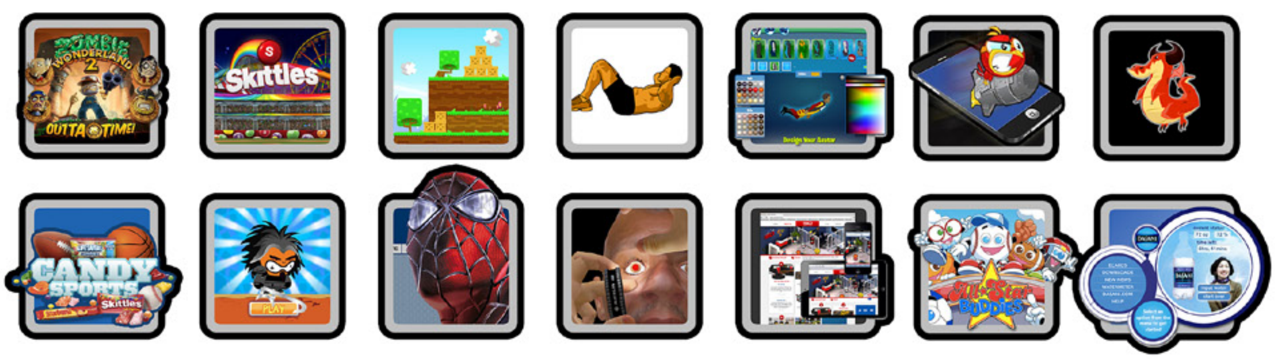
ISO

mobile · interactive · design



Click to View

- ☑ Mobile Apps
- ☑ Website Design
- ☑ Specialty Programming
- ☑ 3DSimulations
- ☑ Unity 3D
- ☑ SmartFoxServer
- ☑ Games
- ☑ Web & Database Dev
- ☑ Super friendly :)



reach out & let's talk: troy@isointeractive.com

www.isointeractive.com

Dr.WEB®
CureIt!™

EMERGENCY CURING

for Windows workstations and servers
including those running other anti-virus software

FUNCTIONS:

- Cures Windows workstations and servers.
- Verifies the quality of the anti-virus software currently in use.

FEATURES:

- Dr.Web CureIt! doesn't require installation and doesn't conflict with any known anti-virus; consequently there is no need to disable the anti-virus currently in use to check a system with Dr.Web CureIt!.
- Improved self-protection and an enhanced mode for more efficient countermeasures against Windows blockers.
- Dr.Web CureIt! is updated at least once an hour.
- The utility can be launched from removable media including USB storage devices.

LICENSING FEATURES:

The utility is available for free when used for non-business purposes.

Doctor Web is the Russian developer of Dr.Web anti-virus software. Dr.Web anti-virus software has been developed since 1992. Doctor Web is one of the few anti-virus vendors in the world to have its own technologies to detect and cure malware. Dr.Web anti-virus software allows IT environments to effectively withstand any threats, even those not yet known.